(54) Title: REDUCING MOTION ARTIFACTS IN NMR IMAGES
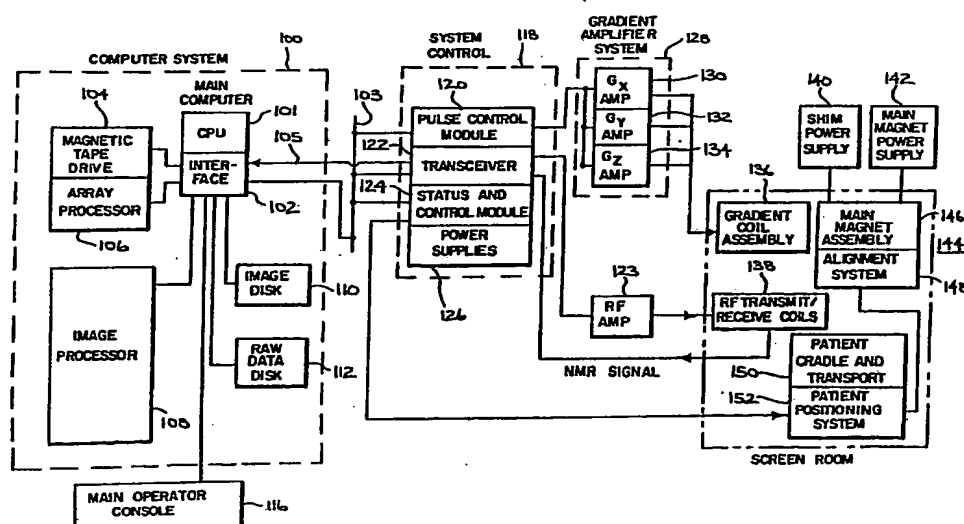
(57) Abstract

An NMR data set (310, 311) which is acquired at (218) with applied phase encoding and read-out gradients at (136) is corrected to reduce motion artifacts and increase image sharpness. The acquired NMR data set (310, 311) is examined at (100) to detect bulk displacements of the object being imaged and phase displacements caused by motion. This information is employed to produce correction operators (330, 331, 337) which are applied to the NMR image data set. One or more navigator signals (340) may also be acquired at (218) during the scan to produce an NMR data set from which the corrective operators can more readily be derived.

.1

REDUCING MOTION ARTIFACTS IN NMR IMAGES

Cross Reference

This application is a continuation-in-part of U.S. Application Serial No. 07/276,168, filed November 23, 1988.

5 ## Background of the Invention

The field of the invention is nuclear magnetic resonance imaging methods and systems. More particularly, the invention relates to a method for reducing image artifacts caused by flow and motion.

10   Any nucleus which possesses a magnetic moment attempts to align itself with the direction of the magnetic field in which it is located. In doing so, however, the nucleus precesses around this direction at a characteristic angular frequency (Larmor frequency)

15   which is dependent on the strength of the magnetic field and on the properties of the specific nuclear species (the magnetogyric constant γ of the nucleus). Nuclei which exhibit this phenomena are referred to herein as "spins".

20   When a substance such as human tissue is subjected to a uniform magnetic field (polarizing field B_z), the individual magnetic moments of the spins in the tissue attempt to align with this polarizing field, but precess about it in random order at their characteristic Larmor

25   frequency. A net magnetic moment M_z is produced in the

direction of the polarizing field, but the randomly
oriented magnetic components in the perpendicular, or
transverse, plane (x-y plane) cancel one another. If,
however, the substance, or tissue, is subjected to a
5 magnetic field (excitation field B_1) which is in the x-y
plane and which is near the Larmor frequency, the net
aligned moment, M_z, may be rotated, or "tipped", into
the z-y plane to produce a net transverse magnetic
moment M_1, which is rotating, or spinning, in the x-y
10 plane at the Larmor frequency. The degree to which the
net magnetic moment M_z is tipped, and hence the
magnitude of the net transverse magnetic moment M_1
depends primarily on the length of time and the
magnitude of the applied excitation field B_1.

15   The practical value of this phenomenon resides in
the signal which is emitted by the excited spins after
the excitation signal B_1 is terminated. In simple
systems the excited spin induce an oscillating sine wave
signal in a receiving coil. The frequency of this
20 signal is the Larmor frequency, and its initial
amplitude, A_0, is determined by the magnitude of the
transverse magnetic moment M_1. The amplitude, A, of the
emission signal decays in an exponential fashion with
time, t:

25            $$A = A_0 e^{-t/T^*_2}$$

     The decay constant $1/T^*_2$ depends on the homogeneity
of the magnetic field and on T_2, which is referred to as

3

the "spin-spin relaxation" constant, or the "transverse
relaxation" constant. The T_2 constant is inversely
proportional to the exponential rate at which the
aligned precession of the spins would dephase after
5   removal of the excitation signal B_1 in a perfectly
homogeneous field.

Another important factor which contributes to the
amplitude A of the NMR signal is referred to as the
spin-lattice relaxation process which is characterized
10   by the time constant T_1. This is also called the
longitudinal relaxation process as it describes the
recovery of the net magnetic moment M to its equilibrium
value along the axis of magnetic polarization (z). The
T_1 time constant is longer than T_2, much longer in most
15   substances of medical interest.

The NMR measurements of particular relevance to the
present invention are called "pulsed NMR measurements".
Such NMR measurements are divided into a period of
excitation and a period of signal emission. Such
20   measurements are performed in a cyclic manner in which
the NMR measurement is repeated many times to accumulate
different data during each cycle or to make the same
measurement at different locations in the subject. A
wide variety of preparative excitation techniques are
25   known which involve the application of one or more
excitation pulses (B_1) of varying magnitude and
duration. Such excitation pulses may have a narrow
frequency spectrum (selective excitation pulse), or they

4

may have a broad frequency spectrum (nonselective

excitation pulse) which produces transverse

magnetization M_1 over a range of resonant frequencies.

The prior art is replete with excitation techniques that

5   are designed to take advantage of particular NMR

phenomena and which overcome particular problems in the

NMR measurement process. The present invention may be

used with any of these pulse sequences.

When utilizing NMR to produce images, a technique

10   is employed to obtain NMR signals from specific

locations in the subject. Typically, the region which

is to be imaged (region of interest) is scanned by a

sequence of NMR measurement cycles which vary according

to the particular localization method being used. The

15   resulting set of received NMR signals are digitized and

processed to reconstruct the image using one of many

well known reconstruction techniques. To perform such a

scan, it is, of course, necessary to elicit NMR signals

from specific locations in the subject. This is

20   accomplished by employing magnetic fields (G_x, G_y, and

G_z) which have the same direction as the polarizing

field B_0, but which have a gradient along the respective

x, y and z axes. By controlling the strength of these

gradients during each NMR cycle, the spatial

25   distribution of spin excitation can be controlled and

the location of the resulting NMR signals can be

identified.

5

NMR data for constructing images can be collected

using one of many available techniques, such as multiple

angle projection reconstruction and Fourier transform

(FT). Typically, such techniques comprise a pulse

5   sequence made up of a plurality of sequentially

implemented views. Each view may include one or more

NMR experiments, each of which comprises at least an RF

excitation pulse and a magnetic field gradient pulse to

encode spatial information into the resulting NMR

10 signal. As is well known, the NMR signal may be a free

indication decay (FID) or, preferably, a spin-echo

signal.

The preferred embodiments of the invention will be

described in detail with reference to a variant of the

15 well known FT technique, which is frequently referred to

as "spin-warp". The spin-warp technique is discussed in

an article entitled "Spin Warp NMR Imaging and

Applications to Human Whole-Body Imaging" by W.A.

Edelstein et al., Physics in Medicine and Biology, Vol.

20 25, pp. 751-756 (1980).

Briefly, the spin-warp technique employs a variable

amplitude phase encoding magnetic field gradient pulse

prior to the acquisition of NMR spin-echo signals to

phase encode spatial information in the direction of

25 this gradient. In a two-dimensional implementation

(2DFT), for example, spatial information is encoded in

one direction by applying a phase encoding gradient (G_y)

along that direction, and then a spin-echo signal is

6

acquired in the presence of a read-out magnetic field

gradient (G_x) in a direction orthogonal to the phase

encoding direction. The read-out gradient present

during the spin-echo acquisition encodes spatial

5   information in the orthogonal direction. In a typical

2DFT pulse sequence, the magnitude of the phase encoding

gradient pulse G_y is incremented (ΔG_y) in the sequence

of views that are acquired during the scan to produce a

set of NMR data from which an entire image can be

10   reconstructed.

Object motion during the acquisition of NMR image

data produces both blurring and "ghosts" in the phase-

encoded direction. Ghosts are particularly apparent

when the motion is periodic, or nearly so. For most

15   physiological motion each view of the NMR signal is

acquired in a period short enough that the object may be

considered stationary during the acquisition window. In

such case the blurring and ghosting is due to the

inconsistent appearance of the object from view to view.

20   Motion that changes the appearance between views such as

that produced by a patient moving, by the respiration or

the cardiac cycle, or by peristalsis, is referred to

hereinafter as "view-to-view motion". Motion may also

change the amplitude and phase of the NMR signal as it

25   evolves during the pulse sequence and such motion is

referred to hereinafter as "in-view motion".

Both blurring and ghosting can be reduced if the

data acquisition is synchronized with the functional

cycle of the object to reduce view-to-view motion. This
method is known as gated NMR scanning, and its objective
is to acquire NMR data at the same point during
successive functional cycles so that the object "looks"

5   the same in each view. The drawback of gating is that
NMR data may be acquired only during a small fraction of
the object's functional cycle, and even when the
shortest acceptable pulse sequence is employed, the
gating technique can significantly lengthen the data

10   acquisition time. Some of these methods are disclosed
in U.S. Patent Nos. 4,751,462; 4,567,893 and 4,663,591.
None of them have proven entirely satisfactory because
they either depend upon perfectly periodic motion, or
they increase the scan time significantly, or they

15   produce low signal-to-noise images.

Several NMR pulse sequences have been proposed to
either desensitize the NMR measurement to the phase
perturbations caused by flowing spins as described in
U.S. Patent No. 4,728,890, or to sensitize it to flow in

20   such a manner that the effects of flow can properly be
separated from the reconstructed images as described in
U.S. Patent No. RE 32,701. None of these methods have
proven entirely satisfactory, either from a performance
standpoint, or because of their adverse impact on scan

25   time or the type of NMR measurements that may be
performed.

In our prior U.S. Patent No. 4,715,383, we disclose
a method for reducing motion and flow artifacts in NMR

images. While this method substantially improves NMR images by suppressing artifacts caused by spins outside the region of interest, it does not correct for the motion artifacts produced by spins located inside the

5 region of interest.

All of the prior methods for reducing motion and flow artifacts focus on the data acquisition procedure. They change the NMR pulse sequence itself, they change the order in which the pulse sequences in a scan are

10 executed, or they synchronize the execution of the pulse sequence with the motion of the subject under study. Their objective is to produce a set of NMR data which is minimally affected by flow and motion and which can, therefore, be used to construct a clear, ghost-free

15 image.

Summary of the Invention

The present invention is a method for reducing motion and flow artifacts in an NMR image by correcting the set of NMR data which has been acquired during a

20 scan to remove the effects of motion and flow before the image is reconstructed. More specifically, the invention includes transforming the NMR data set to create a hybrid-space data array; producing a correction data array using the data in the hybrid-space data

25 array; and applying the data in the correction data array to the NMR data set produced by the NMR system to reduce flow and motion artifacts in the image which is

reconstructed from the NMR data set. The correction
data array which is produced in accordance with one
aspect of the present invention corrects for view-to-
view motion artifacts and the correction data array
5   which is produced in accordance with another aspect of
the invention corrects for in-view motion and flow
artifacts.

     A general object of the invention is to provide a
motion and flow artifact correction method which can be
10   employed after the NMR data has been acquired. The
present invention enables the NMR data set to be
corrected retrospective of its acquisition and,
therefore, it may be used in addition to any motion or
flow artifact suppression techniques which have been
15   employed in the past. To the extent that the acquired
NMR data is affected by motion or flow, the present
invention will detect it and automatically correct the
NMR data so that the affect is reduced or eliminated.

     Another aspect of the invention is to acquire
20   "navigator" NMR data along with the usual image NMR data
within the same pulse sequence. The navigator NMR data
enables the corrections for view-to-view and in-view
motion artifacts to be made more accurately. A
navigator NMR signal is produced in each pulse sequence
25   along with the image NMR signal and a data set is
acquired for both. The corrective values are determined
using the navigator data set and the corrections are
made to the image data set.

10

A more specific object of the invention is to
correct the NMR data set for view-to-view artifacts
caused by motion or flow in any direction. The
navigator signal can be acquired in the presence of a

5 read-out magnetic field gradient which is oriented in
any direction. The shift corrections which are produced
according to the present invention, will correct for
errors caused by view-to-view motion or flow in the
direction of the navigator signal read-out gradient. By

10 acquiring more than one navigator signal in the pulse
sequence in the presence of read-out gradients oriented
in respective different directions, shift corrections
are produced which correct for view-to-view motion in
the corresponding directions. For example, shift

15 corrections can be made for motion along both the x axis
and the y axis of the NMR system.

Yet another specific object of the invention is to
correct the NMR data set for in-view artifacts caused by
motion or flow in any direction. The phase corrections

20 P which are produced according to the present invention
offset artifact causing systematic noise produced by in-
view motion or flow regardless of its direction.

Still another aspect of the present invention is
that both in-view and view-to-view motion and flow

25 artifact correction can be applied to the same NMR data
set. Both the in-view correction data array and the
view-to-view correction data array can be applied to the
NMR data set.

The foregoing and other objects and advantages of
the invention will appear from the following
description. In the description, reference is made to
the accompanying drawings which form a part hereof, and

5   in which there is shown by way of illustration a
preferred embodiment of the invention. Such embodiment
does not necessarily represent the full scope of the
invention, however, and reference is made therefore to
the claims herein for interpreting the scope of the

10   invention.

Brief Description of the Drawings

Fig. 1 is a block diagram of an NMR system which
employs the present invention;

Fig. 2 is an electrical block diagram of the

15   transceiver which forms part of the NMR system of Fig.
1;

Fig. 3 is a graphic representation of a
conventional NMR pulse sequence used to acquire data to
produce an image;

20   Fig. 4 is a pictorial representation of how an
image is reconstructed from NMR data acquired using the
pulse sequence of Fig. 3;

Fig. 5 is a pictorial representation of how
correction values are calculated according to the

25   present invention;

Fig. 6 is a graphic representation of data in the
modulus array of Fig. 5;

Fig. 7 is a graphic representation of the correlation process used to produce the shift correction array of Fig. 5;

Figs. 8a and 8b are graphic representations of data

5 in the phase array of Fig. 5 and the process used to produce the phase correction array of Fig. 5;

Fig. 9 is a graphic representation of an alternative pulse sequence used to acquire both image NMR data and navigator NMR data; and

10 Fig. 10 is a plot of hybrid-space data which illustrates the effects of both stationary and moving spins on hybrid-space phase.

Description of the Preferred Embodiment

Referring to Fig. 1, there is shown in block diagram form the major components of a preferred NMR

15 system which incorporates the present invention and which is sold by the General Electric Company under the trademark "SIGNA". The overall operation of the system is under the control of a host computer system generally designated 100 which includes a main computer 101 (a

20 Data General MV4000). The computer 100 includes an interface 102 through which a plurality of computer peripheral devices and other NMR system components are coupled to the main computer 101. Among the computer peripheral devices is a magnetic tape drive 104 which

25 may be utilized under the direction of the main computer 101 for archiving patient data and image data to tape.

Processed patient data may also be stored in an image
disc storage device designated 110. An array processor
106 is utilized for preprocessing acquired NMR data and
for image reconstruction. The function of image

5   processor 108 is to provide interactive image display
manipulation such as magnification, image comparison,
gray-scale adjustment and real time data display. The
computer system 100 also includes a means to store raw
NMR data (i.e. before image construction) which employs

10   a disc data storage system designated 112. An operator
console 116 is also coupled to the main computer 101 by
means of interface 102, and it provides the operator
with the means to input data pertinent to a patient
study as well as additional data necessary for proper

15   NMR system operation, such as calibrating, initiating
and terminating scans. The operator console is also
used to display images stored on disc or magnetic tape.

   The computer system 100 exercises control over the
NMR system by means of a system control 118 and a

20   gradient amplifier system 128. Under the direction of a
stored program, the computer 100 communicates with
system control 118 by means of a serial communication
network 103 (such as the Ethernet network) in a manner
well known to those skilled in the art. The system

25   control 118 includes several subsystems such as a pulse
control module (PCM) 120, a radio frequency transceiver
122, a status control module (SCM) 124, and power
supplies generally designated 126. The PCM 120 utilizes

14

control signals generated under program control by main
computer 101 to generate digital waveforms which control
gradient coil excitation, as well as RF envelope
waveforms utilized in the transceiver 122 for modulating
5   the RF excitation pulses. The gradient waveforms are
applied to the gradient amplifier system 128 which is
comprised of G_x, G_y and G_z amplifiers 130, 132 and 134,
respectively. Each amplifier 130, 132 and 134 is
utilized to excite a corresponding gradient coil in an
10   assembly designated 136 which is part of a magnet
assembly 146. When energized, the gradient coils
generate magnetic field gradients G_x, G_y and G_z.

The gradient magnetic fields are utilized in
combination with radio frequency pulses generated by
15   transceiver 122, RF amp 123 and RF coil 138 to encode
spatial information into the NMR signals emanating from
the region of the patient being studied. Waveforms and
control signals provided by the pulse control module 120
are utilized by the transceiver subsystem 122 for RF
20   carrier modulation and mode control. In the transmit
mode, the transmitter provides a radio frequency signal
to an RF power amplifier 123 which then energizes RF
coils 138 which are situated within main magnet assembly
146. The NMR signals radiated by the excited spin in
25   the patient are sensed by the same or a different RF
coil than is used for transmitting. The signals are
detected, amplified, demodulated, filtered, and
digitized in the receiver section of the transceiver

122. The processed signals are transmitted to the main

computer 101 by means of a dedicated, unidirectional,

high-speed digital link 105 which links interface 102

and transceiver 122.

5   The PCM 120 and SCM 124 are independent subsystems

both of which communicate with main computer 101,

peripheral systems, such as patient positioning system

152, as well as to one another by means of serial

communications link 103. The PCM 120 and SCM 124 are

10   each comprised of a 16-bit microprocessor (such as Intel

8086) for processing commands from the main computer

101. The SCM 124 includes means for acquiring

information regarding patient cradle position, and the

position of the moveable patient alignment light fan

15   beam (not shown). This information is used by main

computer 101 to modify image display and reconstruction

parameters. The SCM 124 also initiates functions such

as actuation of the patient transport and alignment

systems.

20   The gradient coil assembly 136 and the RF transmit

and receiver coils 138 are mounted within the bore of

the magnet utilized to produce the polarizing magnetic

field. The magnet forms a part of the main magnet

assembly which includes the patient alignment system

25   148, a shim coil power supply 140, and a main magnet

power supply 142. The main power supply 412 is utilized

to bring the polarizing field produced by the magnet to

16

the proper operating strength of 1.5 Tesla and is then
disconnected.

To minimize interference from external sources, the
NMR system components comprised of the main magnet

5   assembly, the gradient coil assembly, and the RF
transmit and receiver coils, as well as the patient-
handling devices, are enclosed in an RF shielded room
generally designated 144. The shielding is generally
provided by a copper or aluminum screen network which

10 encloses the entire room. The screen network serves to
contain the RF signals generated by the system, while
shielding the system from RF signals generated outside
the room.

Referring particularly to Figs. 1 and 2, the

15 transceiver 122 includes components which produce the RF
excitation field B_1 through power amplifier 123 at a
coil 138A and components which receive the resulting NMR
signal induced in a coil 138B. The base, or carrier,
frequency of the RF excitation field is produced by a

20 frequency synthesizer 200 which receives a set of
digital signals through the communications link 103 from
the main computer 101. These digital signals indicate
the frequency which is to be produced at an output 201
at a resolution of one Hertz. This commanded RF carrier

25 is applied to a modulator 202 where it is frequency and
amplitude modulated in response to signals received
through line 203, and the resulting RF excitation signal
is turned on and off in response to a control signal

which is received from the PCM 120 through line 204.
The magnitude of the RF excitation pulse output through
line 205 is attenuated by a transmit attenuator circuit
206 which receives a digital signal from the main

5   computer 101 through communications link 103. The
attenuated RF excitation pulses are applied to the power
amplifier 123 that drives the RF transmitter coil 138A.

Referring still to Figs. 1 and 2, the NMR signal
produced by the excited spin in the subject is picked up

10   by the receiver coil 138B and applied to the input of a
receiver 207. The receiver 207 amplifies the NMR signal
and this is attenuated by an amount determined by a
digital attenuation signal received from the main
computer 101 through link 103. The receiver 207 is also

15   turned on and off by a signal through line 208 from the
PCM 120 such that the NMR signal is acquired only over
the time intervals required by the particular
acquisition being performed.

The received NMR signal is demodulated by a

20   quadrature detector 209 to produce two signals I and Q
that are coupled through anti-aliasing filters 216 and
217 to a pair of analog to digital converters indicated
collectively at 218. The quadrature detector 209 also
receives an RF reference signal from a second frequency

25   synthesizer 210 and this is employed by the quadrature
detector 209 to sense the amplitude of that component of
the NMR signal which is in phase with the transmitter RF
carrier (I signal) and the amplitude of that component

18

of the NMR signal which is in quadrature therewith (Q signal).

The I and Q components of the received NMR signal are continuously sampled and digitized by the A/D

5   converter 218 at a sample rate of 64 kHz throughout the acquisition period. A set of 256 digital numbers are acquired for each I and Q component of the NMR signal, and these digital numbers are conveyed to the main computer 101 through the serial link 105.

10   The NMR system of Fig. 1 performs a series of pulse sequences to collect sufficient NMR data to reconstruct an image. One such pulse sequence is shown in Fig. 3. This sequence performs a slice selection by applying a 90° selective RF excitation pulse 300 in the presence of

15   a z axis gradient pulse 301 and its associated rephasing pulse 302. After an interval $TE_1/2$, a 180° selective RF excitation pulse 303 is applied in the presence of another z axis gradient pulse 304 to refocus the transverse magnetization at the time TE_1 and produce an

20   echo NMR signal 305.

To position encode the echo NMR signal 305, an x axis read-out gradient pulse 306 is applied during the acquisition of the NMR signal 305. The read-out gradient frequency encodes the NMR signal 305 in the

25   well known manner. In addition, the echo NMR signal 305 is position encoded along the y axis by a phase encoding gradient pulse 307. The phase encoding gradient pulse 307 has one strength during each echo pulse sequence and

19

associated NMR echo signal 305, and it is typically

incremented in steps through 256 discrete strengths (-

128 to +128) during the entire scan. As a result, each

of the 256 NMR echo signals 305 acquired during the scan

5   is uniquely phase encoded.

It is, of course, usual practice to repeat the

pulse sequence for each phase encoding gradient value

one or more times and to combine the acquired NMR

signals in some manner to improve signal-to-noise and to

10   offset irregularities in the magnetic fields. In the

following discussion, it is assumed that such techniques

may be used to acquire the NMR data set which is to be

corrected.

Referring particularly to Fig. 4, the acquired NMR

15   data is stored in the data disk 112 (Fig. 1) in the form

of two 256 X 256 element arrays 310 and 311. The array

310 contains the in-phase magnitude values I and the

array 311 contains the quadrature values Q. Together

these arrays 310 and 311 form an NMR image data set

20   which defines the acquired image in what is referred to

in the art as "k-space".

To convert this k-space NMR data set into data

which defines the image in real space (i.e. Cartesian

coordinates), a two step Fourier transformation is

25   performed on the I and Q arrays 310 and 311. The

transformation is performed first in the read-out

direction which is the horizontal rows of the arrays 310

and 311 to produce two 256 X 256 element arrays 312 and

313. The array 312 contains the in-phase data and is
labeled I', while the array 313 contains the quadrature
data and is labeled Q'. The I' and Q' arrays 312 and
313 define the acquired image in what is referred to in
5 the art as "hybrid-space". This first transformation of
the acquired NMR data set is expressed mathematically as
follows:

$$S_{xy}(\text{k-space}) \quad \Rightarrow \quad S_{x'y} \text{ (hybrid-space)}$$

$$S_{x'y} = \frac{1}{256} \sum_{x=0}^{255} S_{xy}e^{-i2\pi xx'/256} \qquad (1)$$

10   The second transformation is performed in the phase
encoding direction which is the vertical columns of the
arrays 312 and 313 to produce two 256 X 256 element
arrays 314 and 315. The array 314 contains the
transformed in-phase values and is labeled I", while the
15 array 315 contains the quadrature values and is labeled
Q". This second transformation may be expressed
mathematically as follows:

$$S_{x'y}(\text{hybrid-space}) \quad \Rightarrow \quad S_{x'y'} \text{ (real space)}$$

$$S_{x'y'} = \frac{1}{256} \sum_{y=0}^{255} S_{x'y}e^{-i2\pi yy'/256} \qquad (2)$$

20 The arrays 314 and 315 are a data set which defines the
acquired image in real space and the elements thereof
are used to calculate the intensity values in a 256 X

21

256 element image array 316 in accordance with the

following expression:

$$Im_{xy} = \sqrt{(I''_{xy})^2 + (Q''_{xy})^2} \qquad\qquad (3)$$

The 256 X 256 elements of the image array 316 are mapped

5   to the main operator console 116 (Fig. 1) for display on

a CRT screen.

The above described NMR system and pulse sequence

is representative of the current state of the art. The

diagnostic quality of the image which is obtained is

10   determined by the extent to which the acquired NMR

signals are degraded by superimposed thermal

(statistical) noise and systematic (artifact) noise

(which includes unsharpness). The sources of thermal

noise are well understood. The ratio of NMR signal

15   intensity to thermal noise is determined by such factors

as polarizing magnetic field strength, RF receiver coil

configuration, the particular pulse sequence used, and

the amount of signal averaging used. Systematic noise,

mainly resulting from physiological motion, may degrade

20   the NMR image far beyond the fundamental limit set by

the thermal noise. Indeed, the diagnostic quality of

many NMR images is limited far more by motion artifact

and other forms of systematic noise than by intrinsic

thermal noise. However, in contrast to thermal noise,

25   there is no inherent limit on the extent to which

systematic noise can be reduced.

It is the reduction of systematic noise and
unsharpness which the present invention addresses.
Whereas prior techniques address this problem with
improvements to the NMR system hardware (for example,

5   cardiac and respiratory gating circuits) or with
improvements to the pulse sequence (for example, motion
desensitizing gradient pulses), the present invention
typically addresses the problem retrospectively. More
specifically, it is a discovery of the present invention

10 that systematic noise can be easily detected by
examining the NMR data set in hybrid-space. Systematic
noise caused by view-to-view motion as well as
systematic noise caused by in-view motion can be
detected. Corrective operators can then be calculated

15 and used to eliminate the systematic noise from the NMR
image data set. The usual image reconstruction process
can then be performed using the corrected NMR image data
set to produce an image which is substantially free of
artifacts caused by motion.

20   Referring to Fig. 5, the first step is to produce
the hybrid-space I' and Q' arrays 312 and 313 as
described above (equation 1). The image data set itself
can be used to produce the hybrid-space arrays 312 and
313, or as will be described in more detail below,

25 separate NMR data produced by a navigator NMR signal
within the same pulse sequence can be used. In either
case, corrective values for both view-to-view and in-

23

view systematic noise can be calculated from the hybrid-space arrays.

Referring to Figs. 5 and 6, it has been discovered that view-to-view systematic noise can be detected in the magnitude information contained in the hybrid-space arrays 312 and 313. Accordingly, the next step in the process is to produce a 256 X 256 modulus array (M) 320. This is accomplished by calculating each element M_{xy} of the modulus array 320 from the corresponding elements I'_{xy} and Q'_{xy} of the hybrid-space arrays 312 and 313.

$$M_{xy} = \sqrt{(I'_{xy})^2 + (Q'_{xy})^2} \qquad\qquad (4)$$

where: x = array column number, 1-256; and

           y = array row number, 1-256.

It is a discovery of the present invention that view-to-view motion in the direction of the associated read-out gradient can be seen as a shifting left or right of the modulus data in each row of the modulus array 320. This is illustrated in Fig. 6, where each graph is the modulus values in a horizontal row of the modulus array 320 plotted as a function of its column number in the array 320. Although each graph is slightly different due to the differences in the amount of phase encoding, the graphs do have significant peaks which should occur at the same column number of the array 320. This is illustrated by the dashed lines 325 and 326 through the peaks 327 and 328 in the graph of the first row (y = 1). The corresponding peaks 327' and

24

328' in the graph of the second row (y = 2) are shifted to the left of the dashed lines 325 and 326, and the corresponding peaks 327" and 328" in the last row of data (y = 256) are shifted to the right. Examination of

5   the remaining 509 rows of the modulus array 320 would show similar shifts in varying amounts which are caused by view-to-view movement of the subject.

The next step in the process is to determine the amount, S, by which each row of modulus data must be

10   shifted to bring it into correlation with a reference row. This cross correlation may be performed in a number of ways. In the preferred embodiment a reference row (M_r) in the modulus array 320 is selected and it is cross correlated with each of the other 255 rows M_y.

15   This is done by determining how much each row M_y must be shifted left or right to maximize the sum of the product of its elements and the corresponding elements in the reference row M_r.

In other words, the following expression is

20   calculated and stored:

$$\sum_{x=1}^{256} M_{rx} \ (M_{yx}) \tag{5}$$

The data on row M_y is then shifted one position and the process is repeated. These calculations are performed for M_y shifted between -32 and +32 positions,

25   although these can be extended if necessary for larger displacements. A plot of an exemplary correlation curve

25

which results from these sixty-four calculations is
shown in Fig. 7. It can be seen that the peak in this
exemplary curve occurs when the row of modulus data M_y
is shifted to the left a few positions. Accordingly, a
5   corrective value S_y is determined by finding the number
of shifts needed to produce the peak in the correlation
curve. A shift correction value S_y is calculated for
each row (y = 1 to 256) of the modulus array 320 and is
stored in a 1 X 256 element shift correction array 330.
10 There are many other ways to correlate each row in the
modulus array 320.

A very significant reduction in view-to-view motion
in the x axis can be achieved by applying the correction
values S directly to the modulus array 320 and
15 reconstructing an image from this corrected NMR data.
This is accomplished by shifting the data in each row of
the modulus 320 by the amount indicated by the
corresponding element of the shift correction array 330.
Then, the values in the I' and Q' hybrid-space arrays
20 312 and 313 are calculated using the corrected modulus
values and the known phase angle for each element. An
image can then be reconstructed in the usual fashion
from the corrected I' and Q' hybrid-space arrays 312 and
313 as described above.

25   While the above-described correction procedure
provides significant improvement in image quality,
further improvements can be made. When the shift
correction S is made, a small amount of phase error is

26

introduced into the NMR data set. This is due to the
phase rollover which occurs in any NMR system as a
result of asymmetry in the echo signal in its
acquisition window. The signal represented by the
5   elements in each column of the I' and Q' hybrid arrays
312 and 313 contain the same amount of rollover phase,
and the amount of rollover phase changes linearly at the
rollover rate K_R as the arrays are traversed from left
to right through its columns (x = 1 to 256). Thus, when
10   data in a row is shifted left or right to make the S
correction discussed above, the data moves into a column
with a different phase value than the column of its
origin. This rollover phase can be corrected and is a
function of the amount which the data was shifted:

15       $\Phi R = K_R * S.$

The rollback correction values for each row in
hybrid-space is calculated to produce a 1 X 256 element
rollback correction array 331. Each element in the
array 331 indicates the phase correction which must be
20   made to all elements in its corresponding row of the I'
and Q' hybrid-space arrays 312 and 313. How these phase
corrections are actually made will be discussed in more
detail below.

The correction made thus far accounts for view-to-
25   view motion along the read-out x axis. As will be
discussed in more detail below, similar corrections can
be made along the phase encoding y axis by using a

27

special navigator signal in the same pulse sequence as
the image signal and which is acquired in the presence
of a y axis read-out gradient pulse.

The above-described corrections do not account for
5   phase errors in the NMR data due to in-view motion or
flow. Such errors occur because the spins are moving
during the pulse sequence. It is a further discovery of
the present invention that these phase errors can be
detected in the k-space or the hybrid-space NMR data
10   set.

Referring to Figs. 5 and 8, the procedure begins
again with the I' and Q' hybrid-space arrays 312 and
313. This data set is used to calculate the phase Φ of
the acquired NMR signal at each of the 256 sample times
15   and at each of the 256 views. A 256 by 256 element
phase array 335 is produced, with each of its elements
having a value which is calculated as follows:

$$\Phi_{xy} = TAN^{-1} (Q'_{xy}/I'_{xy})$$

It is a discovery of the present invention that the
20   calculated phase Φ should have the same value along any
column (y = 1 to 256) of the phase array 335. This is
shown graphically in Fig. 8a where the calculated phase
Φ for three rows of data has been plotted as a function
of column number (x = 1 to 256). For any given NMR data
25   set which is produced without phase encoding gradients
(i.e. a navigator signal), these plots will be
substantially the same if there are no phase errors. To

28

the extent that the calculated phase Φ differs in any column, that difference represents a phase error which should be corrected. This is graphically illustrated in Fig. 8b where two of the phase plots in Fig. 8a are

5   superimposed on one another to reveal differences in their values over the central region. These differences are illustrated by the arrows Φ_1 and Φ_2.

To correct for the phase error, therefore, a 256 by 256 element phase correction array 337 is produced.

10 This is accomplished by establishing one row (y = 1) of phase data in the Φ array 335 as a reference and then finding the difference between the value of one of its elements and the same element in each of the other rows (y = 2 to 256) of the Φ array 335. This is repeated for

15 each element (x = 1 to 256) of the reference row (y = 1) and the calculated difference values ($\Delta\Phi_p$) are stored in the corresponding location in the phase correction array 337. Thus, each value in the phase correction array 337 indicates the amount by which the phase of

20 each element of the NMR data set should be corrected. This corrects for in-view motion in any direction (x, y or z). How this correction is actually made will be described in more detail below.

While separate phase correction values $\Delta\Phi_p$ are

25 calculated for each of the 256 by 256 elements of the array 337 in the preferred embodiment, it should be apparent that a less rigorous approach can also be used. If the in-view motion is known to occur over only a

29

small segment of the x axis field of view, then the

calculation of corrective values $\Delta\Phi_P$ may be limited to

that segment. Furthermore, if the in-view motion is

uniform over that x axis segment, then perhaps a single

5   value $\Delta\Phi_P$ will suffice as a correction over the entire

segment. Since these corrections are applied

retrospectively to the acquired NMR data set, it is

contemplated that correction variables such as this will

be under operator control and the radiologist can

10   manipulate the corrections to obtain the image he needs

in minimal time.

When calculating the phase corrections which are to

be made, one must consider whether both moving and

stationary spins are contributing to the measured phase

15   values. Where all the spins are moving, no further

correction is necessary. However, when stationary spins

are contributing substantial signal, then the phase

correction values can be further refined.

The effects of NMR signal components produced by

20   both stationary and moving spins is illustrated in Fig.

10. The points on this plot represent the I' and Q'

values for the 256 data samples in a single column of

the hybrid-space data sets 312 and 313. These points

define a circle which is illustrated by dashed line 401

25   having a center 402. If no motion were present, all of

the points in this ring would cluster at one location.

On the other hand, if all the spins are in motion, then

the ring 401 would be formed, but its center 402 would

be at the origin $(Q'=0, I'=0)$. In the illustrated
example, the signal components due to stationary spins
are represented by the vector A which offsets the center
402 from the origin. The signal component due to moving

5   spins is represented by the vector B and its magnitude
determines the size of the circle 401. The values (Φ)
in the phase array 335 discussed above indicate the
phase angle of the combined stationary and moving spins.
To provide a more accurate correction for in-view

10   motion, therefore, the values in the phase array 335
should be altered to indicate the phase Φ' of only the
moving spins.

       This alteration of the phase array values is
performed on one column at a time. First, the 256 data

15   points in the column of the hybrid-space arrays 312 and
313 are applied to a curve fitting program which
determines the center 402 of the circle 401. The
coordinate values of the center 402 are then subtracted
from the values in that column of the respective hybrid-

20   space arrays 312 and 313. These altered values are then
used to produce the altered phase array 335, which, in
turn, is used to produce the more accurate phase
correction array 337 as described above.

       While the phase error detection method described

25   above works well when applied to a hybrid-space phase
array which is derived from data acquired from navigator
NMR signals, less rigorous phase error detection methods
are also possible using the NMR image data set itself.

For example, the I and Q arrays 310 and 311 can be used
to calculate a 256 X 256 phase array in k-space. While
the phase values in this array may not be useful at its
outer boundaries, the phase information at the center
5   column (x=128) where the peak of the NMR echo signal 305
is sampled is usually unambiguous. One element in this
column is selected as the reference and all the other
values are compared with it to produce a 1 X 256 element
phase correction array. These correction values $\Delta\Phi_P$ may
10 be applied to the NMR data set as will now be described.

All of the corrections calculated according to the
present invention can be made to the NMR raw image data
set I and Q in k-space. In k-space all of the
corrections are implemented as a rotation of the phase
15 of each NMR signal sample ($S_{xy} = I_{xy} + jQ_{xy}$). The
corrected NMR data set S'_{xy} may thus be calculated as
follows:

$$S'_{xy} = e^{i\Delta\Phi_T} S_{xy} \tag{6}$$

This rotation is implemented in the I and Q arrays
20 310 and 311 in accordance with the following
expressions:

$$I_{xy} \text{ (corrected)} = I_{xy}\cos\Delta\Phi_T - Q_{xy}\sin\Delta\Phi_T$$

$$Q_{xy} \text{ (corrected)} = I_{xy}\sin\Delta\Phi_T + Q_{xy}\cos\Delta\Phi_T$$

The phase change $\Delta\Phi_T$ is the arithmetic sum of the
25 phase changes needed to make all of the above-described

corrections. Specifically, the total phase correction $\Delta\Phi_T$ is calculated as follows:

$$\Delta\Phi_T = -(\Delta\Phi_R + K * \Delta\Phi_P + \Delta\Phi_x + \Delta\Phi_y) \tag{7}$$

5   This expression includes the rollback correction $\Delta\Phi_R$ from the array 331 (Fig. 5) and the phase correction $\Delta\Phi_P$ from the array 337 (Fig. 5). The phase correction $\Delta\Phi_P$ is multiplied by a conversion factor K, however, to account for any difference in phase shifts between the image NMR signal and a navigator NMR signal which may be

10   used to determine the phase shift correction. If the image NMR data set is used to calculate the phase corrections, this conversion factor is "one". Otherwise, the value of K is measured by comparing the phase shift produced in image data (with phase encoding

15   gradient applied) and the phase shift produced in the navigator data. In the alternative, the value of K can be calculated.

    The phase rotations $\Delta\Phi_x$ and $\Delta\Phi_y$ are made to correct for view-to-view motion and flow effects along

20   the respective x and y axes. These phase corrections are calculated from the shift values S in the shift correction array 330 (Fig. 5):

$$\Delta\Phi_x = S_x * (x-(N_x-1)/2) * 2\pi/N_x \tag{9}$$

where: S_x = shift correction from array 330

25   x = sample number (i.e. 0 through 255)

                 N_x = total number of samples during read-out (256).

33

$$\Delta\Phi_y = S_y * (y-(N_y-1)/2) * 2\pi/N_y \qquad\qquad (10)$$

where: S_y = shift correction calculated from
navigator   signal acquired in
presence of a y axis   magnetic field
gradient

y = phase encoding view number (0 to
255)

N_y = total number of phase encoding
views   (256).

It should be apparent that the correction is
different for each element of the NMR image data arrays
I and Q because the total phase correction $\Delta\Phi_T$ is
different for each x and y location in these arrays.

As indicated above, the corrective methods of the
present invention are applied to the NMR data set which
has been acquired for the purpose of producing the
desired image. While this same image data set can also
be used to derive the corrective values which are to be
applied to it, an alternative procedure is to produce a
separate NMR data set during the same acquisition scan
from which the corrective values can more accurately be
derived. This separate NMR data set is produced by
acquiring one or more "navigator" NMR signals during
each pulse sequence. The main distinction of these
navigator NMR signals is that they are not subject to
the application of incremented phase encoding gradients.
In some cases, a constant phase encode gradient may be
used to elicit two-dimensional investigatory
information.

34

A navigator pulse sequence is shown in Fig. 9. As with the conventional pulse sequence of Fig. 3, the spins are excited by a 90° selective RF excitation pulse 300 in the presence of slice select gradient pulses 301

5   and 302, and at a time interval $TE_1/2$ later, the spins are subjected to a 180° selective RF excitation pulse 303 in the presence of slice select gradient pulse 304. No phase encoding gradients are applied, however, and at interval TE_1 an NMR echo signal 340 is acquired in the

10 presence of x axis read-out gradient pulse 306. A second 180° RF excitation pulse 341 is then produced in the presence of a slice select gradient pulse 342, and the y axis phase encoding pulse 307 is then applied. The resulting NMR echo signal 305 produced at TE_2 is

15 then acquired in the presence of read-out gradient pulse 306' to build the NMR image data set as described above.

The NMR data set which is created by acquiring the NMR navigator signal 340 is subject to substantially the same motion effects as the NMR image data set which is

20 acquired at the same time. However, because the navigator signals 340 are not phase encoded, the magnitude of the navigator NMR signals 340 are not significantly diminished at the extremities of the field of view. Corrective values can, therefore, be more

25 readily calculated using the NMR navigator signal data set rather than the NMR image data set. The corrective values described above are thus determined by using the acquired NMR navigator signal data set, and these

35

corrective values are then applied to the NMR image data

set. The corrected NMR image data set is then used to

reconstruct an image.

Referring again to Fig. 5, the NMR navigator signal

5   data set is transformed first along the read-out

gradient direction to produce the hybrid-space I' and Q'

arrays 312 and 313. This data is then used as described

above to produce a shift correction array 330, a

rollback correction array 331 and a phase correction

10 array 337. These corrective values are applied to the

NMR image data set 310 and 311 (Fig. 4) as described

above, and the corrected NMR image data set is processed

in the usual manner to produce the corrected image array

316 (Fig. 4).

15   The shift corrections S correct for view-to-view

motion in the direction of the read-out gradient.

Accordingly, the NMR navigator signal 340 in Fig. 9 is

acquired in the presence of the read-out gradient pulse

306 which is directed along the x axis and the shift

20 corrections will compensate for motion along the x axis.

Alternatively, the NMR navigator signal 340 can also be

acquired in the presence of a read-out gradient pulse

which is directed along the phase encoding, or y axis,

as shown by the dashed lines 343. In yet another

25 alternative, the pulse sequence can be modified to

produce two NMR navigator signals, with one being

acquired during an x axis read-out gradient pulse and

the other being acquired in the presence of a y axis

36

read-out gradient pulse. Either one of the navigator

signal NMR data sets can then be used to produce the

phase correction array 337 (Fig. 5), while both

navigator signal NMR data sets are used to produce

5   corresponding shift correction arrays 330. A rollback

correction array 331 is not always required for shift

corrections made in the phase encoding direction, y.

Both the x and y shift corrections can be made in

hybrid-space, or they may be converted to phase angle

10 corrections which are made to the NMR image data set in

k-space as described above.

The two-dimensional shift correction data can also

be derived using a single navigator NMR signal acquired

during the application of a read-out gradient. As

15 previously described, the magnitude of the navigator NMR

signal yields the correction values S along the

frequency encoding direction (x). In addition, however,

the phase of this navigator NMR signal can be used to

calculate the correction values along the phase encoding

20 direction (y) as follows:

$$\frac{\Delta \Phi_y}{N_y} = \frac{\Delta \Phi_{py}}{A_y} \qquad (11)$$

where: $\Delta \Phi_{py}$ is the measured phase shift, A_y is

the amplitude of the applied phase

encoding gradient, and N_y is the number of

25   phase encoding increments (i.e. views).

37

The correction values for use in the above equation (10)
for the y direction is, therefore, given by the
following when N_y is 256:

$$\Delta\Phi_y = \frac{\Delta\Phi_{py}(256)}{2\pi(j-128.5)} \qquad (12)$$

5   where: j is the view number

While the navigator NMR signal is an echo signal in
the preferred embodiment, this is not a necessary
requirement. The navigator NMR signal may also be
produced as a gradient recalled signal or as the result
10 of three-dimensional data acquisition pulse sequence.
The preferred embodiment of the above-described
invention is implemented by a Fortran program which is
executed by the main computer 101 for correcting in-view
motion and view-to-view motion in the direction of the x
15 axis read-out gradient. A listing of this program is
provided in Appendix A.
While the corrective values which are derived
according to the present invention are employed
retrospectively to improve image quality, they can also
20 be employed to alter the scan. This is particularly
true of the shift values S which correct for view-to-
view motion. More specifically, as each pulse sequence
is executed and a row of raw NMR data is acquired for
the image arrays 310 and 311, the row of data may be
25 transformed to hybrid-space (equation (1)), the modulus
values calculated (equation (4)), and a correlation made

38

with the previously acquired row of NMR data (equation

(5)). The resulting shift value S can then be used to

alter the operating conditions of the NMR system before

the next pulse sequence is executed. For example, where

5   the shift value corrects for motion in the y axis

direction, the phase of the RF carrier signal produced

by the transmit frequency synthesizer 200 (Fig. 2) is

changed by the amount $-\Delta\Phi_y$ (equation (9)). Similarly, a

shift correction for motion along the slice select

10   direction (z axis) can be made by changing the frequency

of the RF carrier produced by the transmit frequency

synthesizer 200. Shift corrections for motion along the

read-out direction (x axis) can be made by changing the

location of the field of view along the x axis.

15   The present invention addresses the problems

currently associated with magnetic resonance

angiography. In addition, this invention is necessary

for quantitative tissue characterization using magnetic

resonance. The invention is not limited to Fourier

20   image reconstructions (i.e. 2DFT or 3DFT) and may be

used with line scanning and other projection

reconstructions.

39

APPENDIX A

This is the main MARCO program. It gets input from the user, such things
as which file to work with. It then calls the routine DO_MARCO that
does the actual work of getting values from the raw data file. These
values include those for both x direction and intraview motion cor-
rection. These values are then writtng to a text file to be read by the
POLO program.

```
***************************************************************
************ BEGINNING OF MARCO.F77 SOURCE FILE *************
***************************************************************

!
!              COPYRIGHT - MAYO FOUNDATION FOR MEDICAL EDUCATION AND RESEARCH
!                                    1988
          IMPLICIT NONE
          CHARACTER*128 SOURCE_FILE  ! Name of source file
          CHARACTER*128 ASCII_PATH   ! Name of file for ascii output
          INTEGER*4 IERR             ! Error flag
          INTEGER*4 ICHAN1           ! Channel number for source file

          LOGICAL*2 CHOPPER, SHIM, HNEX, STRIP, D3FFT, GOT_IT
          LOGICAL*2 EXORCIST, HI_LO_SORT, CHOPLETSON, NOAP, NO_SORT
          INTEGER*2 SKIP920

          INTEGER*4 IBYTES1

          INTEGER*2 BASELINE_SKIP
          INTEGER*2 FRAMES_PER_SLICE
          INTEGER*2 DATA_BLOCKS
          INTEGER*2 VECTORC              ! Counter into VECTOR, below
          INTEGER*2 NEX
          INTEGER*2 NAVS
          INTEGER*2 NECHOES
          INTEGER*2 FRAME_SIZE
          INTEGER*2 THE_ECHO1            ! Echo from which to get displacement
C                                        ! VECOTR
          INTEGER*4 DELTAS(1024)
          REAL PHASES(1024)
C
C
          INTEGER*2 VIEW_FROM
          INTEGER*2 STARTO

          INTEGER*4 III, I

          LOGICAL*2 DO_DELTAS, DO_ROLL, DO_SHIFT
          INTEGER*2 NAV_ECHO, IMAGE_ECHO
          CHARACTER ANS, ANS1, ANS2, ANS3
          INTEGER*4 GET_I
          INTEGER*2 NEX_IN_RAW
          INTEGER*2 START_DELTA
          INTEGER*2 STOP_DELTA
          INTEGER*2 FRAME_DELTA
          INTEGER*2 START_ROLL
          INTEGER*2 STOP_ROLL
          INTEGER*2 PHASE_COL
          REAL ANGLE_ROLL
          INTEGER*2 PHASE_FRAME
```

```
          WRITE(6,142)
142       FORMAT('1',//////)

          III = 0
          GOT_IT = .FALSE.
          DO WHILE ((.NOT. GOT_IT) .AND. (III .LT. 10))
            GOT_IT = .TRUE.
            III = III + 1
            WRITE(6,3010)
3010        FORMAT('+ENTER SOURCE FILE-----------------------------
-----------<GT>',S)
            READ(*,'(A)') SOURCE_FILE

            DO I= 1 , 128
              IF(SOURCE_FILE(I:I).EQ."<40>") SOURCE_FILE(I:I)="<0>"
            END DO
            CALL LIB$GOPEN (SOURCE_FILE,0,ICHAN1,IBYTES1,IERR)
            IF (IERR.NE.0) THEN
              IF (IERR.EQ.21) THEN
                PRINT *, "SOURCE FILE DOES NOT EXIST."
                PRINT*,CHAR(7)
              ELSE IF (IERR .EQ. 20) THEN
                PRINT*,'ILLEGAL FILENAME CHARACTER.'
                PRINT*,CHAR(7)
              ELSE IF (IERR .EQ. 172) THEN ! ZERO LENGTH FILENAME
                CONTINUE
              ELSE
                PRINT*,'ERROR IN OPENING SOURCE FILE, CODE:',IERR
                PRINT*,CHAR(7)
              END IF
              GOT_IT = .FALSE.
            END IF
          END DO

          IF (III .GE. 10) GO TO 9000

C...
C...
C...      THE FOLLOWING READS PARTS OF THE RAW DATA FILE HEADER TO DE-
          TERMINE CERTAIN THINGS ABOUT THE RAW DATA FILE.

          CALL GET_COLLECTION(ICHAN1,CHOPPER,SHIM,HNEX,STRIP,D3FFT,
    #            EXORCIST,HI_LO_SORT,CHOPLETSON,NOAP,NO_SORT,
    #            BASELINE_SKIP,SKIP920,FRAMES_PER_SLICE,DATA_BLOCKS,
    #            FRAME_SIZE,NECHOES,NAVS,IERR)
          IF (IERR .NE. 0) THEN
            PRINT*,'ERROR FROM GETTING COLLECTION TYPE, CODE:',IERR
            GO TO 9000
          END IF

          IF ((CHOPPER) .AND. (.NOT. NOAP)) THEN
            NEX_IN_RAW = 1
          ELSE IF ((CHOPPER) .AND. (NOAP)) THEN
            NEX_IN_RAW = NAVS * 2
          END IF

          DO I = 1 , 1024
            DELTAS(I) = -42
```

41

```
        PHASES(I)   = -42
        END DO

        DO_DELTAS = .TRUE.
        FRAME_DELTA = -42
        DO WHILE ((FRAME_DELTA .LT. 0) .OR.
     #            (FRAME_DELTA .GT. FRAMES_PER_SLICE * NEX_IN_RAW))
          WRITE(6,4020)FRAMES_PER_SLICE*NEX_IN_RAW
4020      FORMAT('+WHICH OF THE ',I3,' FRAMES DO YOU WANT TO USE FOR THE RE
FERENCE?-------<GT>',$)
          FRAME_DELTA = INT2(GET_I())
        END DO
        START_DELTA = -42
        DO WHILE ((START_DELTA .LT. 0) .OR.
     #            (START_DELTA .GT. FRAME_SIZE))
          WRITE(6,4010)
4010      FORMAT('+WHERE DO YOU WANT TO START TO GET SELECTIVE DELTAS?-----
------------<GT>',$)
          START_DELTA = INT2(GET_I())
        END DO
        IF (START_DELTA .EQ. 0) THEN
          START_DELTA = 1
          STOP_DELTA = FRAME_SIZE
        ELSE
          STOP_DELTA = -42
          DO WHILE ((STOP_DELTA .LT. 0) .OR.
     #              (STOP_DELTA .GT. FRAME_SIZE))
            WRITE(6,4011)
4011        FORMAT('+WHERE DO YOU WANT TO STOP GETTING SELECTIVE DELTAS?---
--------------<GT>',$)
            STOP_DELTA = INT2(GET_I())
          END DO
          IF (STOP_DELTA .LT. START_DELTA) THEN
            I = STOP_DELTA
            STOP_DELTA = START_DELTA
            START_DELTA = I
          END IF
        END IF

        DO_ROLL = .TRUE.

        DO_SHIFT = .TRUE.

        PHASE_FRAME = -42
        DO WHILE ((PHASE_FRAME .LT. 0) .OR.
     #            (PHASE_FRAME .GT. FRAMES_PER_SLICE * NEX_IN_RAW))
          WRITE(6,4030)FRAMES_PER_SLICE*NEX_IN_RAW
4030      FORMAT('+WHICH OF THE ',I3,' FRAMES DO YOU WANT TO USE FOR THE PH
ASE REFERENCE--<GT>',$)
          PHASE_FRAME = INT2(GET_I())
        END DO

        NAV_ECHO = 0
        IMAGE_ECHO = 0

        DO WHILE ((NAV_ECHO .LT. 1) .OR. (NAV_ECHO .GT. NECHOES))
          WRITE(6,4003)NECHOES
4003      FORMAT('+OF THE ',I1,' ECHOES, WHICH IS THE NAVAGATION ECHO?-----
```

42

```
-----------------<GT>',$)
        NAV_ECHO = INT2(GET_I())
     END DO

     DO WHILE ((IMAGE_ECHO .LT. 1) .OR. (IMAGE_ECHO .GT. NECHOES))
        WRITE(6,4004)NECHOES
4004    FORMAT('+OF THE ',I1,' ECHOES, WHICH IS THE IMAGE ECHO?----------
-----------------<GT>',$)
        IMAGE_ECHO = INT2(GET_I())
     END DO

     IF (CHOPLETSON) THEN
        NEX = 1
     ELSE
        NEX = NAVS * 2
     END IF

     IF (HNEX) THEN
        PRINT*,'UNABLE TO PROCESS HALF NEX DATA.'
        GO TO 9000
     END IF

     IF (.NOT. ((CHOPPER) .OR. (CHOPLETSON) .OR. (NOAP))) THEN
        PRINT*,'NOT CHOPPER, NOAP, OR CHOPLETSON, UNABLE TO PROCESS.'
        GO TO 9000
     END IF

     I = 1
     DO WHILE ((I .LT. 122) .AND. (SOURCE_FILE(I:I) .NE. CHAR(0)))
        I = I + 1
     END DO
     IF (I .LT. 122) THEN
        IF ((SOURCE_FILE(I-4:I-1) .EQ. '.RAW') .OR.
   #        (SOURCE_FILE(I-4:I-1) .EQ. '.raw')) THEN
           STARTO = I-4
        ELSE
           STARTO = I
        END IF

        ASCII_PATH = SOURCE_FILE(1:STARTO-1)//'.CNG'
        DO I = STARTO+4 , 128
           ASCII_PATH(I:I) = CHAR(0)
        END DO
     ELSE
        PRINT*,'ERROR, SOURCE FILENAME IS TOO LONG.'
        GO TO 9000
     END IF

     OPEN(UNIT=42,FILE=ASCII_PATH,STATUS='FRESH',IOSTAT=IERR)
     IF(IERR.NE.0) THEN
        PRINT *,'ERROR DETECTED DURING FILE OPEN, CODE : ',IERR
        GO TO 9000
     END IF

     IF (CHOPPER .AND. NOAP) THEN
        WRITE(6,3040)
     ELSE IF (CHOPPER .AND. .NOT. NOAP) THEN
        WRITE(6,3050)
```

43

```
        END IF

        WRITE(6,3060)FRAMES_PER_SLICE,FRAME_SIZE
        WRITE(6,3070)NECHOES,NEX
3040    FORMAT(' SOURCE FILE: CHOPPER AND NOAP')
3050    FORMAT(' SOURCE FILE: CHOPPER AND AP')
3060    FORMAT(' FRAMES PER SLICE = ',I3,T32,'FRAME SIZE = ',I3)
3070    FORMAT(' ECHOES           = ',I3,T32,'NEX          = ',I3,/)


        CALL DO_MARCO(DO_DELTAS,DO_ROLL,DO_SHIFT,FRAME_DELTA,
     #          START_DELTA,STOP_DELTA,START_ROLL,STOP_ROLL,ANGLE_ROLL,
     #          PHASE_COL,PHASE_FRAME,DELTAS,PHASES,NEX_IN_RAW,NECHOES,
     #          FRAME_SIZE,FRAMES_PER_SLICE,DATA_BLOCKS,
     #          ICHAN1,NAV_ECHO,IMAGE_ECHO,IERR)
        VECTORC = FRAMES_PER_SLICE * NEX_IN_RAW
        IF (IERR .NE. 0) THEN
          PRINT*,'ERROR FROM DO_MARCO, CODE=',IERR
          GO TO 9000
        END IF

        WRITE(42,356)DO_DELTAS,FRAME_DELTA,START_DELTA,STOP_DELTA,
     #          DO_ROLL,START_ROLL,STOP_ROLL,ANGLE_ROLL,DO_SHIFT,
     #          PHASE_COL,PHASE_FRAME,IMAGE_ECHO,NAV_ECHO
356     FORMAT('INFO:',L2,I4,I4,I4,L2,I4,I4,F10.4,L2,I4,I4,2(I4))
        DO I = 1 , VECTORC
          WRITE(42,4242)I,DELTAS(I),PHASES(I)
4242      FORMAT(I3,I6,F20.15)
        END DO

        CLOSE(42,IOSTAT=IERR)

9000    CONTINUE ! ERROR EXIT

        END
```

```
******************************************************************
*************** END OF MARCO.F77 SOURCE FILE ******************
******************************************************************


******************************************************************
************ BEGINNING OF DO_MARCO.F77 SOURCE FILE ************
******************************************************************


!
! COPYRIGHT - MAYO FOUNDATION FOR MEDICAL EDUCATION AND RESEARCH - 1988
!
        SUBROUTINE DO_MARCO(DO_DELTAS,DO_ROLL,DO_SHIFT,FRAME_DELTA,
     #          START_DELTA,STOP_DELTA,START_ROLL,STOP_ROLL,ANGLE_ROLL,
     #          PHASE_COL,PHASE_FRAME,DELTAS,PHASES,NEX_IN_RAW,NECHOES,
     #          FRAME_SIZE,FRAMES_PER_SLICE,DATA_BLOCKS,ICHAN1,NAV_ECHO,
     #          IMAGE_ECHO,IERR)
        IMPLICIT NONE
        INTEGER*4 IERR            ! ERROR FLAG
        INTEGER*4 DELTAS(*)       ! ARRAY OF DELTA SHIFTS
        REAL PHASES(*)            ! ARRAY OF PHASE SHIFTS
```

```
        REAL PI9
        PARAMETER (PI9 = 2.827433388) ! PI * 0.9

C...
C...    THE NEXT THREE ARE BOOLEANS THAT DETERMINE WHAT VALUES ARE TO BE
C...    OBTAINED.
        LOGICAL*2 DO_DELTAS       ! TO GET THE DELTA SHIFTS
        LOGICAL*2 DO_ROLL         ! TO GET THE ROLL BACK ANGLE
        LOGICAL*2 DO_SHIFT        ! TO GET THE PHASE SHIFTS

        INTEGER*2 FRAME_DELTA
        INTEGER*2 START_DELTA
        INTEGER*2 STOP_DELTA
        INTEGER*2 START_ROLL
        INTEGER*2 STOP_ROLL
        INTEGER*2 PHASE_COL
        INTEGER*2 NEX_IN_RAW
        INTEGER*2 PHASE_FRAME
        REAL ANGLE_ROLL

        INTEGER*2 BUFFER(16384) ! BUFFER FOR READING IN VIEWS, IF THERE
C                               ! WERE 512 SAMPLES IN A FRAME, FOUR NEX,
C                               ! AND FOUR ECHOES, 16384 WORDS WOULD BE
C                               ! READ
        INTEGER*2 ZBUF(512)
        INTEGER*4 STEP
        INTEGER*4 LN
        INTEGER*4 I, J, K, L2, L, M
        INTEGER*4 ICHAN1
        INTEGER*4 GET_I
        REAL    ROLLER(4,512) ! REAL, IMAG, MOD, ANGLE
        REAL Exiyi, Exi, Eyi, Exi2

        INTEGER*2 FRAMES_PER_SLICE
        INTEGER*2 DATA_BLOCKS
        INTEGER*2 STOP_BLOCK
        INTEGER*2 FRAME_SIZE
        INTEGER*2 NECHOES
        INTEGER*2 NAVS
        INTEGER*2 FOFF

        INTEGER*2 FRAME_FROM
        INTEGER*2 NEX_FROM
        INTEGER*2 CHOPPER_FAC
        INTEGER*2 PHASE_FRAME_FROM
        INTEGER*2 PHASE_NEX_FROM

        INTEGER*2 LOW_CON
        PARAMETER(LOW_CON = -32)
        INTEGER*2 HIGH_CON
        PARAMETER(HIGH_CON = 32)
        INTEGER*2 ONE
        PARAMETER(ONE = 1)
        INTEGER*2 TWO
        PARAMETER(TWO = 2)

        REAL MOD_MAX
        REAL MOD_25
```

```
REAL MOD_10
INTEGER*4 MAXOV1
INTEGER*4 B_C
INTEGER*2 VECTORC
INTEGER*2 PHASEC
REAL RR
REAL RI
REAL VIEW1(-256:256)
REAL VIEWN(-256:256)
REAL THE_MODS(-256:256)
REAL MOD_MEANS(LOW_CON:HIGH_CON)
INTEGER*2 LOW_HALF
INTEGER*2 HIGH_HALF
INTEGER*2 NAV_ECHO
INTEGER*2 IMAGE_ECHO
REAL PHASE_REF

COMPLEX C1(512)
REAL B1(2,512)
EQUIVALENCE(C1,B1)

DO I = 1 , FRAME_SIZE
   ROLLER(1,I) = 0.0
   ROLLER(2,I) = 0.0
END DO

LOW_HALF  = 1-FRAME_SIZE/2
HIGH_HALF = FRAME_SIZE/2

IF (NEX_IN_RAW .EQ. 1) THEN
   FRAME_FROM = FRAME_DELTA-1
   NEX_FROM = 1
   PHASE_FRAME_FROM = PHASE_FRAME-1
   PHASE_NEX_FROM = 1
ELSE
   FRAME_FROM = (FRAME_DELTA - 1) / NEX_IN_RAW
   NEX_FROM = MOD2(FRAME_DELTA-ONE,NEX_IN_RAW)
   PHASE_FRAME_FROM = (PHASE_FRAME - 1) / NEX_IN_RAW
   PHASE_NEX_FROM = MOD2(PHASE_FRAME-ONE,NEX_IN_RAW)
END IF

WRITE(6,2020)

IF (DO_DELTAS) THEN
   K = 1

   DO WHILE (K .LT. START_DELTA)
      ZBUF(K) = 0
      K = K + 1
   END DO

   DO WHILE (K .LE. STOP_DELTA)
      ZBUF(K) = 1
      K = K + 1
   END DO

   DO WHILE (K .LE. FRAME_SIZE)
      ZBUF(K) = 0
```

46

```
         K = K + 1
         END DO

         DO I = -256 , 256
            VIEW1(I) = 0.0
            VIEWN(I) = 0.0
         END DO

      END IF ! DO_DELTAS

      VECTORC = 0
      PHASEC = 0

      FOFF = FRAME_SIZE * 2

      IF (FRAME_SIZE .EQ. 512) THEN
         STEP = 4 * NECHOES * NEX_IN_RAW
         LN = 9
      ELSE IF (FRAME_SIZE .EQ. 256) THEN
         STEP = 2 * NECHOES * NEX_IN_RAW
         LN = 8
      ELSE
         IERR = 4201
         RETURN
      END IF

      STOP_BLOCK = 28 + STEP * FRAMES_PER_SLICE

      IF (STEP * FRAMES_PER_SLICE .GT. DATA_BLOCKS) THEN
         PRINT*,'ERROR, NOT ENOUGH DATA BLOCKS.'
         IERR = 4200
         RETURN
      END IF

      IF (DO_ROLL) THEN
         DO I = 1 , 0 , -1
            CALL LIB$RDB (28+STEP*(FRAME_SIZE/2-I),STEP,BUFFER,ICHAN1,IERR)
            IF(IERR.NE.0) THEN
               IF ((IERR .NE. -1) .AND. (IERR .NE. 24)) THEN
                  PRINT*,' '
                  PRINT *, 'ERROR DETECTED WHEN READING BLOCK NUMBER: ',28+(S
      TEP*FRAME_SIZE/2-I),' CODE:',IERR
               ELSE
                  PRINT*,' '
                  PRINT*,'ERROR, END OF FILE.'
               END IF
               RETURN
            END IF

            DO L2 = 0 , NEX_IN_RAW - 1
               B_C = FOFF * ((IMAGE_ECHO-1) + L2 * NECHOES)

               IF (MOD4(L2,2) .EQ. 1) THEN
                  CHOPPER_FAC = -1
               ELSE
                  CHOPPER_FAC = 1
               END IF
```

47

```
      L = FRAME_SIZE / 2 + 1
      DO J = 1+B_C , FRAME_SIZE+B_C , 2
         C1(L) = CMPLX(BUFFER(J  ) * CHOPPER_FAC ,
   #                   BUFFER(J+1) * CHOPPER_FAC)
         L = L + 1
      END DO

      L = 1
      DO J = FRAME_SIZE+1+B_C , FRAME_SIZE*2+B_C , 2
         C1(L) = CMPLX(BUFFER(J  ) * CHOPPER_FAC ,
   #                   BUFFER(J+1) * CHOPPER_FAC)
         L = L + 1
      END DO

      CALL FFTC(B1(1,1),B1(2,1),LN,2)

      DO J = 1 , FRAME_SIZE
         ROLLER(1,J) = ROLLER(1,J) + AREAL(C1(J))
         ROLLER(2,J) = ROLLER(2,J) + AIMAG(C1(J))
      END DO
   END DO
END DO

MOD_MAX = 0
DO I = 1 , FRAME_SIZE
   ROLLER(1,I) = ROLLER(1,I) / (NEX_IN_RAW * 2)
   ROLLER(2,I) = ROLLER(2,I) / (NEX_IN_RAW * 2)
   ROLLER(3,I) = SQRT(ROLLER(1,I)*ROLLER(1,I)+ROLLER(2,I)*ROLLER(2
,I))

   IF (ROLLER(3,I) .GT. MOD_MAX) THEN
      MOD_MAX = ROLLER(3,I)
   END IF

   IF ((ROLLER(1,I) .EQ. 0) .AND. (ROLLER(2,I) .EQ. 0)) THEN
      ROLLER(4,I) = 0
   ELSE
      ROLLER(4,I) = ATAN2(ROLLER(2,I),ROLLER(1,I))
   END IF
END DO

MOD_25 = MOD_MAX * .25

START_ROLL = 1
STOP_ROLL = -42
DO WHILE ((START_ROLL .LT. FRAME_SIZE - 16) .AND.
   #         (STOP_ROLL .LT. 0))
   DO WHILE ((ROLLER(3,START_ROLL) .LT. MOD_25) .AND.
   #            (START_ROLL .LT. FRAME_SIZE-16))
      START_ROLL = START_ROLL + 1
   END DO

   IF (START_ROLL .LT. FRAME_SIZE - 16) THEN
      IF ((ROLLER(3,START_ROLL+5)  .GE. MOD_25) .AND.
   #       (ROLLER(3,START_ROLL+6)  .GE. MOD_25) .AND.
   #       (ROLLER(3,START_ROLL+7)  .GE. MOD_25) .AND.
   #       (ROLLER(3,START_ROLL+8)  .GE. MOD_25) .AND.
   #       (ROLLER(3,START_ROLL+9)  .GE. MOD_25) .AND.
   #       (ROLLER(3,START_ROLL+10) .GE. MOD_25) .AND.
```

48

```
#              (ROLLER(3,START_ROLL+11) .GE. MOD_25) .AND.
#              (ROLLER(3,START_ROLL+12) .GE. MOD_25) .AND.
#              (ROLLER(3,START_ROLL+13) .GE. MOD_25) .AND.
#              (ROLLER(3,START_ROLL+14) .GE. MOD_25) .AND.
#              (ROLLER(3,START_ROLL+15) .GE. MOD_25) .AND.
#              (ROLLER(4,START_ROLL+5)  .LT. PI9)    .AND.
#              (ROLLER(4,START_ROLL+6)  .LT. PI9)    .AND.
#              (ROLLER(4,START_ROLL+7)  .LT. PI9)    .AND.
#              (ROLLER(4,START_ROLL+8)  .LT. PI9)    .AND.
#              (ROLLER(4,START_ROLL+9)  .LT. PI9)    .AND.
#              (ROLLER(4,START_ROLL+10) .LT. PI9)    .AND.
#              (ROLLER(4,START_ROLL+11) .LT. PI9)    .AND.
#             (ROLLER(4,START_ROLL+12) .LT. PI9)    .AND.
#              (ROLLER(4,START_ROLL+13) .LT. PI9)    .AND.
#              (ROLLER(4,START_ROLL+14) .LT. PI9)    .AND.
#              (ROLLER(4,START_ROLL+15) .LT. PI9)) THEN
            START_ROLL = START_ROLL + 5
            STOP_ROLL = START_ROLL + 10
          ELSE
            START_ROLL = START_ROLL + 15
          END IF
        END IF
      END DO
      IF (STOP_ROLL .GT. 0) THEN
        Exiyi  = 0.0
        Exi    = 0.0
        Eyi    = 0.0
        Exi2   = 0.0
        DO J = START_ROLL , STOP_ROLL
          Exiyi  = Exiyi + J * ROLLER(4,J)
          Exi    = Exi   + J
          Eyi    = Eyi   + ROLLER(4,J)
          Exi2   = Exi2  + J * J
        END DO

        IF (11*Exi2 - (Exi * Exi) .NE. 0) THEN
          ANGLE_ROLL = (11*Exiyi - Exi*Eyi) / (11*Exi2 - Exi*Exi)
        ELSE
          ANGLE_ROLL = -999999999
        END IF
      END IF
    END IF ! DO_ROLL

    IF (DO_SHIFT) THEN
      CALL LIB$RDB (28+STEP*PHASE_FRAME_FROM,STEP,BUFFER,ICHAN1,IERR)
      IF(IERR.NE.0) THEN
        IF ((IERR .NE. -1) .AND. (IERR .NE. 24)) THEN
          PRINT*,' '
          PRINT *, 'ERROR DETECTED WHEN READING BLOCK NUMBER: ',28-STEP
*PHASE_FRAME_FROM,' CODE:',IERR
        ELSE
          PRINT*,' '
          PRINT*,'ERROR, END OF FILE.'
        END IF
        RETURN
      END IF

      B_C = FOFF * ((NAV_ECHO-1) + PHASE_NEX_FROM * NECHOES)
```

```
        IF (MOD2(PHASE_NEX_FROM,TWO) .EQ. 1) THEN
          CHOPPER_FAC = -1
        ELSE
          CHOPPER_FAC = 1
        END IF

        L = FRAME_SIZE/2 + 1
        DO J = 1+B_C , FRAME_SIZE+B_C , 2
          C1(L) = CMPLX(BUFFER(J  ) * CHOPPER_FAC,
     #                  BUFFER(J+1) * CHOPPER_FAC)
          L = L + 1
        END DO

        L = 1
        DO J = FRAME_SIZE+1+B_C , FRAME_SIZE*2+B_C , 2
          C1(L) = CMPLX(BUFFER(J  ) * CHOPPER_FAC,
     #                  BUFFER(J+1) * CHOPPER_FAC)
          L = L + 1
        END DO

        CALL FFTC(B1(1,1),B1(2,1),LN,2)

        MOD_MAX = 0
        DO J = 1 , FRAME_SIZE
          ROLLER(1,J) = AREAL(C1(J))
          ROLLER(2,J) = AIMAG(C1(J))
          ROLLER(3,J) = SQRT(ROLLER(1,J)*ROLLER(1,J)+ROLLER(2,J)*ROLLER(2
     ,J))

          IF (ROLLER(3,J) .GT. MOD_MAX) THEN
            MOD_MAX = ROLLER(3,J)
          END IF

          IF ((ROLLER(1,J) .EQ. 0) .AND. (ROLLER(2,J) .EQ. 0)) THEN
            ROLLER(4,J) = 0
          ELSE
            ROLLER(4,J) = ATAN2(ROLLER(2,J),ROLLER(1,J))
          END IF

        END DO

        MOD_10 = MOD_MAX * .10

        WRITE(6,1999)
1999    FORMAT(/,'   HERE ARE PHASE ANGLES CLOSE TO 0',/)

        DO J = 1 , FRAME_SIZE
          IF ((ROLLER(3,J) .GE. MOD_10) .AND.
     #        (ROLLER(4,J) .GT. -0.1) .AND.
     #        (ROLLER(4,J) .LT. 0.1)) THEN
            WRITE(6,2000)J,ROLLER(4,J)
2000        FORMAT('       COL = ',I3,' ANGLE = ',F6.4)
          END IF
        END DO

        WRITE(6,2020)

        PHASE_COL = -42
```

```
        DO WHILE ((PHASE_COL .LT. 1) .OR. (PHASE_COL .GT. FRAME_SIZE))
           WRITE(6,3000)
3000       FORMAT('+ENTER THE COLUMN TO USE FOR THE PHASE SHIFTS---<GT>',$
)
           PHASE_COL = INT2(GET_I())
        END DO

        PHASE_REF = ROLLER(4,PHASE_COL)

      END IF ! DO_SHIFT

      IF (DO_DELTAS) THEN
        CALL LIBSRDB (28+STEP*FRAME_FROM,STEP,BUFFER,ICHAN1,IERR)
        IF(IERR.NE.0) THEN
           IF ((IERR .NE. -1) .AND. (IERR .NE. 24)) THEN
              PRINT*,' '
              PRINT *, 'ERROR DETECTED WHEN READING BLOCK NUMBER: ',I,' COD
E:',IERR
           ELSE
              PRINT*,' '
              PRINT*,'ERROR, END OF FILE.'
           END IF
           RETURN
        END IF

        B_C = FOFF * ((NAV_ECHO-1) + NEX_FROM * NECHOES)

        IF (MOD2(NEX_FROM,TWO) .EQ. 1) THEN
           CHOPPER_FAC = -1
        ELSE
           CHOPPER_FAC = 1
        END IF

        L = FRAME_SIZE/2 + 1
        DO J = 1+B_C , FRAME_SIZE+B_C , 2
           C1(L) = CMPLX(BUFFER(J  ) * CHOPPER_FAC,
     #                   BUFFER(J+1) * CHOPPER_FAC)
           L = L + 1
        END DO

        L = 1
        DO J = FRAME_SIZE+1+B_C , FRAME_SIZE*2+B_C , 2
           C1(L) = CMPLX(BUFFER(J  ) * CHOPPER_FAC,
     #                   BUFFER(J+1) * CHOPPER_FAC)
           L = L + 1
        END DO

        CALL FFTC(B1(1,1),B1(2,1),LN,2)

        L = 1
        DO J = 1 , FRAME_SIZE/2
           RR = AREAL(C1(J))
           RI = AIMAG(C1(J))
           VIEW1(L) = SQRT(RR*RR+RI*RI) * ZBUF(L+HIGH_HALF)
           L = L + 1
        END DO

        L = LOW_HALF
```

```
            DO J = FRAME_SIZE/2 + 1 , FRAME_SIZE
               RR = AREAL(C1(J))
               RI = AIMAG(C1(J))
               VIEW1(L) = SQRT(RR*RR+RI*RI) * ZBUF(L+HIGH_HALF)
               L = L + 1
            END DO
         END IF ! DO_DELTAS

         WRITE(6,2020)

         IF ((DO_DELTAS) .OR. (DO_SHIFT)) THEN
            I = 28
            K = 0
            IERR = 0
            DO WHILE (I .LT. STOP_BLOCK)
               K = K + 1
               CALL LIB$RDB (I,STEP,BUFFER,ICHAN1,IERR)
               IF(IERR.NE.0) THEN

                  IF ((IERR .NE. -1) .AND. (IERR .NE. 24)) THEN
                     PRINT*,' '
                     PRINT *, 'ERROR DETECTED WHEN READING BLOCK NUMBER: ',I,' C
ODE:',IERR

                  ELSE
                     PRINT*,' '
                     PRINT*,'ERROR, END OF FILE.'
                  END IF
                  RETURN
               END IF
               WRITE(6,2010)K
2010           FORMAT('+--<GT>',I3,$)

               IF (DO_DELTAS) THEN
                  DO L2 = 0 , NEX_IN_RAW - 1
                     B_C = FOFF * ((NAV_ECHO-1) + L2 * NECHOES)

                     IF (MOD4(L2,2) .EQ. 1) THEN
                        CHOPPER_FAC = -1
                     ELSE
                        CHOPPER_FAC = 1
                     END IF

                     L = FRAME_SIZE / 2 + 1
                     DO J = 1+B_C , FRAME_SIZE+B_C , 2
                        C1(L) = CMPLX(BUFFER(J  ) * CHOPPER_FAC ,
     #                               BUFFER(J+1) * CHOPPER_FAC)
                        L = L + 1
                     END DO

                     L = 1
                     DO J = FRAME_SIZE+1+B_C , FRAME_SIZE*2+B_C , 2
                        C1(L) = CMPLX(BUFFER(J  ) * CHOPPER_FAC ,
     #                               BUFFER(J+1) * CHOPPER_FAC)
                        L = L + 1
                     END DO

                     CALL FFTC(B1(1,1),B1(2,1),LN,2)
```

52

```
                IF (DO_SHIFT) THEN
                  PHASEC = PHASEC + 1
                  IF ((AREAL(C1(PHASE_COL)) .EQ. 0) .AND.
                      (AIMAG(C1(PHASE_COL)) .EQ. 0)) THEN
                    PHASES(PHASEC) = 0 - PHASE_REF
                  ELSE
                    PHASES(PHASEC) =
                      ATAN2(AIMAG(C1(PHASE_COL)),AREAL(C1(PHASE_COL))) -
                      PHASE_REF
                  END IF
                END IF ! DO_SHIFT

                L = 1
                DO J = 1 , FRAME_SIZE / 2
                  RR = AREAL(C1(J))
                  RI = AIMAG(C1(J))
                  VIEWN(L) = SQRT(RR*RR+RI*RI)
                  L = L + 1
                END DO

                L = LOW_HALF
                DO J = FRAME_SIZE / 2 + 1 , FRAME_SIZE
                  RR = AREAL(C1(J))
                  RI = AIMAG(C1(J))
                  VIEWN(L) = SQRT(RR*RR+RI*RI)
                  L = L + 1
                END DO

                VECTORC = VECTORC + 1

                DO L = LOW_CON , HIGH_CON
                  DO M = LOW_HALF , HIGH_HALF
                    THE_MODS(M) = VIEW1(M+L) * VIEWN(M)
                  END DO
                  MOD_MEANS(L) = 0.0
                  DO M = LOW_HALF , HIGH_HALF
                    MOD_MEANS(L) = MOD_MEANS(L) + THE_MODS(M)
                  END DO
                  MOD_MEANS(L) = MOD_MEANS(L) / 256 ! (HIGH_CON - LOW_CON -
1)
                END DO

                MAXOV1 = -42
                DO L = LOW_CON , HIGH_CON
                  IF (MOD_MEANS(L) .GT. MAXOV1) THEN
                    MAXOV1 = MOD_MEANS(L)
                    DELTAS(VECTORC) = L
                  END IF
                END DO
              END DO ! FOR EACH NEX
            END IF ! DO_DELTAS
            I = I + STEP
          END DO
        END IF ! DO_DELTAS OR DO_SHIFT
2020    FORMAT(' ')
        WRITE(6,2020)

        RETURN
```

```
            END

************************************************************
************* END OF DO_MARCO.F77 SOURCE FILE ****************
************************************************************

Those are the two main components of the MARCO program. The following
two are the main components of the POLO program, the program that
actually changes the raw data to implement the motion crrections
mentioned above. The main POLO program gets some information from
the user and reads the text file created in the MARCO program, and the
routine DO_POLO implements the changes.

************************************************************
************* BEGINNING OF POLO.F77 SOURCE FILE ****************
************************************************************


!   COPYRIGHT - MAYO FOUNDATION FOR MEDICAL EDUCATION AND RESEARCH - 1988
!

c       POLO

c       THIS PROGRAM TAKES THE DATA GENERATED BY MARCO AND APPLIES
c       THEM TO A RAW FILE, WRITTING THE NEW DATA TO A RAW FILE

        IMPLICIT NONE
        CHARACTER*128 SOURCE_FILE  ! Name of source file
        CHARACTER*128 ASCII_PATH   ! Name of file of ascii values
        CHARACTER*128 DESP_PATH    ! Name of file with displacement
c                                  ! correction
        INTEGER*4 IERR             ! Error flag
        INTEGER*4 ICHAN1           ! Channel number for source file
        INTEGER*4 ICHAN2           ! Channel for displacement correction

c...
c...    The following are identifiers that are used with the routine
c...    GET_COLLECTION, which gets some information about the raw data.
c...
        LOGICAL*2 CHOPPER          ! If the file has chopper data
        LOGICAL*2 NOAP             ! If the data has been processed by
c                                  ! array processer
        LOGICAL*2 CHOPLETSON       ! If the file has chopletson data
        LOGICAL*2 SHIM, HNEX, STRIP, D3FFT, GOT_IT
        LOGICAL*2 EXORCIST, HI_LO_SORT, NO_SORT
        INTEGER*2 SKIP920
        INTEGER*2 BASELINE_SKIP    ! Number of blocks of baseline data
        INTEGER*2 FRAMES_PER_SLICE ! Frames per slice
        INTEGER*2 DATA_BLOCKS      ! Number of data blocks
        INTEGER*2 NAVS             ! Number of averages
        INTEGER*2 NECHOES          ! Number of echoes
        INTEGER*2 FRAME_SIZE       ! Frame size


        INTEGER*4 IBYTES           ! Number of bytes in opened file

c...    These next two relate to the ascii file that holds information
c...    from the marco program. If the file was written in an older
```

```
C....    version of the program, only delta values and frame numbers are
C....    on the file. The new verion has other information, and the first
C....    line begins with 'INFO:'

         LOGICAL*2 NEW_VERSION
         CHARACTER*5 IS_THIS_INFO

         INTEGER*2 VECTORC          ! Counter into VECTOR, below
         INTEGER*2 NEX              ! Number of excitations
         INTEGER*2 NEX_IN_RAW       ! Number of excitaions in the raw
C                                   ! file. Data that have been processed
C                                   ! by the array processer has only one
C                                   ! while other chopper data have 2 or 4

C....    The nex two help make the new raw data file's name.
         INTEGER*2 STARTO
         CHARACTER UNIQUE

         INTEGER*4 III, I, K, J     ! counters

C....    The next fifteen store information from the ascii data file.

         INTEGER*4 DELTAS(1024)     ! The delta values, how far to shift the
C                                   ! data
         REAL PHASES(1024)          ! The phase shift in a selected column
         LOGICAL*2 DO_DELTAS        ! To shift by the delta values
         LOGICAL*2 DO_ROLL          ! To do the roll correction
         LOGICAL*2 DO_SHIFT         ! To do phase shift correction
         INTEGER*2 FRAME_DELTA      ! What frame was used as a reference for
C                                   ! getting the deltas values
         INTEGER*2 START_DELTA      ! The sample number at which the
C                                   ! determination of the delta values
C                                   ! began
         INTEGER*2 STOP_DELTA       ! The sample number at which the
C                                   ! determination of the delta values
C                                   ! ended
         INTEGER*2 START_ROLL       ! Column at which roll angle
C                                   ! determination began
         INTEGER*2 STOP_ROLL        ! Column at which roll angle
C                                   ! determination ended
         INTEGER*2 PHASE_COL        ! Column used for phase shift
C                                   ! determination
         INTEGER*2 PHASE_FRAME      ! Frame used for phase shift
C                                   ! determination
         INTEGER*2 IMAGE_ECHO       ! Which echo is the image
         INTEGER*2 NAV_ECHO         ! Which echo is the navigator
         REAL ANGLE_ROLL            ! The roll angle

         REAL*8 DELTA_FACTOR
         REAL*8 PHASE_FACTOR
         INTEGER*4 PARSE_F

         WRITE(6,142)
142      FORMAT('1',//////)

         III = 0
         GOT_IT = .FALSE.
         DO WHILE ((.NOT. GOT_IT) .AND. (III .LT. 10))
```

```
        GOT_IT = .TRUE.
        III = III + 1
        WRITE(6,3010)
3010    FORMAT('+ENTER SOURCE FILE--------------------------------------
-----------<GT>',$)
        READ(*,'(A)') SOURCE_FILE

        DO I= 1 , 128
          IF(SOURCE_FILE(I:I).EQ."<40>") SOURCE_FILE(I:I)="<0>"
        END DO
        CALL LIB$GOPEN (SOURCE_FILE,0,ICHAN1,IBYTES,IERR)
        IF (IERR.NE.0) THEN
          IF (IERR.EQ.21) THEN
            PRINT *, "SOURCE FILE DOES NOT EXIST."
            PRINT*,CHAR(7)
          ELSE IF (IERR .EQ. 20) THEN
            PRINT*,'ILLEGAL FILENAME CHARACTER.'
            PRINT*,CHAR(7)
          ELSE IF (IERR .EQ. 172) THEN ! ZERO LENGTH FILENAME
            CONTINUE
          ELSE
            PRINT*,'ERROR IN OPENING SOURCE FILE, CODE:',IERR
            PRINT*,CHAR(7)
          END IF
          GOT_IT = .FALSE.
        END IF
      END DO

      IF (III .GE. 10) GO TO 9000

C...
C...    THIS GETS SOME INFORMATION ABOUT THE RAW DATA FILE BY READING
C...    PARTS OF THE FILE HEADER.
        CALL GET_COLLECTION(ICHAN1,CHOPPER,SHIM,HNEX,STRIP,D3FFT,
     #          EXORCIST,HI_LO_SORT,CHOPLETSON,NOAP,NO_SORT,
     #          BASELINE_SKIP,SKIP920,FRAMES_PER_SLICE,DATA_BLOCKS,
     #          FRAME_SIZE,NECHOES,NAVS,IERR)
        IF (IERR .NE. 0) THEN
          PRINT*,'ERROR FROM GETTING COLLECTION TYPE, CODE:',IERR
          GO TO 9000
        END IF

        IF ((CHOPPER) .AND. (.NOT. NOAP)) THEN
          NEX_IN_RAW = 1
        ELSE IF ((CHOPPER) .AND. (NOAP)) THEN
          NEX_IN_RAW = 2 * NAVS
        END IF

        III = 0
        GOT_IT = .FALSE.
        DO WHILE ((.NOT. GOT_IT) .AND. (III .LT. 10))
          III = III + 1
          WRITE(6,3015)
3015      FORMAT('+ENTER FILE OF DELTA VALUES------------------------
-----------<GT>',$)
          READ(*,'(A)') ASCII_PATH

          DO I= 1 , 128
```

56

```
        IF(ASCII_PATH(I:I) .EQ. "<40>") ASCII_PATH(I:I)="<0>"
        END DO

        OPEN(UNIT=42,FILE=ASCII_PATH,STATUS='OLD',IOSTAT=IERR)
        IF(IERR .NE. 0) THEN
          PRINT *,'ERROR DETECTED DURING FILE OPEN, CODE : ',IERR
        ELSE
          GOT_IT = .TRUE.
        END IF
        END DO

        IF (III .GE. 10) GO TO 9000

        VECTORC = FRAMES_PER_SLICE
        IF (NOAP) VECTORC = FRAMES_PER_SLICE * NAVS * 2
C...    THIS PART CHECKS TO SEE IF THE FILE OF ASCII DATA IS THE
C...    'NEW' VERSION, OR THE 'OLD' VERSION. THE OLD VERSION ONLY
C...    HAS THE DELTA VALUES AND NO HEADER INFORMATION. THE NEW VERSION
C...    HAS MORE INFORMATION.

        READ(42,4241)IS_THIS_INFO
4241    FORMAT(A5)
        IF (IS_THIS_INFO .EQ. 'INFO:') THEN
          NEW_VERSION = .TRUE.
        ELSE
          NEW_VERSION = .FALSE.
        END IF

        REWIND(42)

        IF (NEW_VERSION) THEN
          READ(42,4200)IS_THIS_INFO,DO_DELTAS,FRAME_DELTA,START_DELTA,
     *         STOP_DELTA,DO_ROLL,START_ROLL,STOP_ROLL,ANGLE_ROLL,
     *         DO_SHIFT,PHASE_COL,PHASE_FRAME,IMAGE_ECHO,NAV_ECHO
4200      FORMAT(A5,L2,3(I4),L2,2(I4),F10.4,L2,2(I4),2(I4))

          DO I = 1 , VECTORC
            READ(42,4243)K,DELTAS(I),PHASES(I)
4243        FORMAT(I3,I6,F20.15)
          END DO
        ELSE ! THIS IS AN OLD VERSION
          DO_DELTAS = .TRUE.
          DO_ROLL = .FALSE.
          DO_SHIFT = .FALSE.
          START_DELTA = -1
          STOP_DELTA = -1
          DO I = 1 , VECTORC
            READ(42,4245)K,DELTAS(I),J
4245        FORMAT(I3,2(I6))
          END DO
        END IF
        CLOSE(42,IOSTAT=IERR)
```

57

```
        IF (DO_DELTAS) THEN
           UNIQUE = '*'
           DO WHILE ((UNIQUE .NE. 'Y') .AND. (UNIQUE .NE. 'N') .AND.
     #               (UNIQUE .NE. 'y') .AND. (UNIQUE .NE. 'n'))
              WRITE(6,4000)
4000          FORMAT('+DO YOU WANT TO DO THE DELTA SHIFT CORRECTION ? (Y/N)--
     ------------<GT>',$)
              READ(5,3999)UNIQUE
3999          FORMAT(A)
           END DO
           IF ((UNIQUE .EQ. 'N') .OR. (UNIQUE .EQ. 'n')) THEN
              DO_DELTAS = .FALSE.
           END IF
           IF (DO_DELTAS) THEN
              WRITE(6,4500)
4500          FORMAT('+ENTER THE DELTA SHIFT FACTOR -----------------------
     -----------<GT>',$)
              K = PARSE_F(1,DELTA_FACTOR)
              IF (K .EQ. 0) DELTA_FACTOR = 1
           END IF
        END IF

        IF (DO_ROLL) THEN
           UNIQUE = '*'
           DO WHILE ((UNIQUE .NE. 'Y') .AND. (UNIQUE .NE. 'N') .AND.
     #               (UNIQUE .NE. 'y') .AND. (UNIQUE .NE. 'n'))
              WRITE(6,4001)
4001          FORMAT('+DO YOU WANT TO DO THE ROLL BACK CORRECTION ? (Y/N) ---
     ------------<GT>',$)
              READ(5,3999)UNIQUE
           END DO

           IF ((UNIQUE .EQ. 'N') .OR. (UNIQUE .EQ. 'n')) THEN
              DO_ROLL = .FALSE.
           END IF
        END IF

        IF (DO_SHIFT) THEN
           UNIQUE = '*'
           DO WHILE ((UNIQUE .NE. 'Y') .AND. (UNIQUE .NE. 'N') .AND.
     #               (UNIQUE .NE. 'y') .AND. (UNIQUE .NE. 'n'))
              WRITE(6,4002)
4002          FORMAT('+DO YOU WANT TO DO PHASE SHIFT CORRECTION ? (Y/N)------
     ------------<GT>',$)
              READ(5,3999)UNIQUE
           END DO

           IF ((UNIQUE .EQ. 'N') .OR. (UNIQUE .EQ. 'n')) THEN
              DO_SHIFT = .FALSE.
           END IF
           IF (DO_SHIFT) THEN
              WRITE(6,4510)
4510          FORMAT('+ENTER THE PHASE SHIFT FACTOR ----------------------
     -----------<GT>',$)
              K = PARSE_F(1,PHASE_FACTOR)
              IF (K .EQ. 0) PHASE_FACTOR = 1
           END IF
        END IF
```

58

```
      IF (CHOPLETSON) THEN
         NEX = 1
      ELSE
         NEX = NAVS * 2
      END IF

      IF (HNEX) THEN
         PRINT*,'UNABLE TO PROCESS HALF NEX DATA.'
         GO TO 9000
      END IF

      IF (.NOT. ((CHOPPER) .OR. (CHOPLETSON) .OR. (NOAP))) THEN
         PRINT*,'NOT CHOPPER, NOAP, OR CHOPLETSON, UNABLE TO PROCESS.'
         GO TO 9000
      END IF

      I = 1
      DO WHILE ((I .LT. 122) .AND. (SOURCE_FILE(I:I) .NE. CHAR(0)))
         I = I + 1
      END DO
      IF (I .LT. 122) THEN
         IF ((SOURCE_FILE(I-4:I-1) .EQ. '.RAW') .OR.
     #       (SOURCE_FILE(I-4:I-1) .EQ. '.raw')) THEN
            IF ((SOURCE_FILE(I-5:I-5) .LT. '0') .OR.
     #          (SOURCE_FILE(I-5:I-5) .GT. '9')) THEN
               STARTO = I-5
            ELSE
               STARTO = I-4
            END IF
         ELSE
            STARTO = I
         END IF

         UNIQUE = '*'
         DO WHILE ((UNIQUE .LT. 'A') .OR.
     #             ((UNIQUE .GT. 'Z') .AND. (UNIQUE .LT. 'a')) .OR.
     #             (UNIQUE .GT. 'z'))
            WRITE(6,3019)
3019        FORMAT('+ENTER CHARCTER TO UNIQUELY IDENTIFY THE OUTPUT FILE---
     ----------------<GT>',$)
            READ(*,'(A)') UNIQUE
         END DO
         DESP_PATH = SOURCE_FILE(1:STARTO-1)//UNIQUE//'.DSP'
         DO I = STARTO+5 , 128
            DESP_PATH(I:I) = CHAR(0)
         END DO
      ELSE
         PRINT*,'ERROR, SOURCE FILENAME IS TOO LONG.'
         GO TO 9000
      END IF

C...   THIS MAKES A NEW FILE FOR THE RAW DATA.

      CALL MAKE_NEW_UDF(DESP_PATH, IERR)
      IF (IERR .NE. 0) THEN
         PRINT *,'ERROR DETECTED DURING DESP FILE MAKE, CODE : ',IERR
         GO TO 9000
```

59

```
        END IF

        CALL LIB$GOPEN (DESP_PATH,0,ICHAN2,IBYTES,IERR)
        IF (IERR .NE. 0) THEN
          PRINT *,'ERROR DETECTED DURING DESP FILE OPEN, CODE : ',IERR
          GO TO 9000
        END IF

        IF (CHOPPER .AND. NOAP) THEN
          WRITE(6,3040)
        ELSE IF (CHOPPER .AND. .NOT. NOAP) THEN
          WRITE(6,3050)
        END IF

        WRITE(6,3060)FRAMES_PER_SLICE,FRAME_SIZE
        WRITE(6,3070)NECHOES,NEX

3040    FORMAT(' SOURCE FILE: CHOPPER AND NOAP')
3050    FORMAT(' SOURCE FILE: CHOPPER AND AP')
3060    FORMAT(' FRAMES PER SLICE = ',I3,T32,'FRAME SIZE = ',I3)
3070    FORMAT(' ECHOES           = ',I3,T32,'NEX        = ',I3,/)

C...    DO_POLO IS THE ROUTINE WHERE ALL THE NUMBER WORK IS DONE

        CALL DO_POLO(DELTAS,PHASES,DO_DELTAS,DO_ROLL,DO_SHIFT,
     #         START_DELTA,STOP_DELTA,ANGLE_ROLL,NEX_IN_RAW,NECHOES,
     #         FRAME_SIZE,FRAMES_PER_SLICE,DATA_BLOCKS,IMAGE_ECHO,
     #         ICHAN1,ICHAN2,DELTA_FACTOR,PHASE_FACTOR,IERR)
        IF (IERR .NE. 0) THEN
          PRINT*,'ERROR FROM DO_POLO, CODE=',IERR
        END IF

9000    CONTINUE ! GONE TO FOR ERRORS

        END


********************************************************************
****************** END OF POLO.F77 SOURCE FILE ******************
********************************************************************


********************************************************************
*************** BEGINNING OF DO_POLO.F77 SOURCE FILE ************
********************************************************************


!
!       COPYRIGHT - MAYO FOUNDATION FOR MEDICAL EDUCATION AND RESEARCH
!
        SUBROUTINE DO_POLO(DELTAS,PHASES,DO_DELTAS,DO_ROLL,DO_SHIFT,
     #         START_DELTA,STOP_DELTA,ANGLE_ROLL,NEX_IN_RAW,NECHOES,
     #         FRAME_SIZE,FRAMES_PER_SLICE,DATA_BLOCKS,IMAGE_ECHO,
     #         ICHAN1,ICHAN2,DELTA_FACTOR,PHASE_FACTOR,IERR)
        IMPLICIT NONE
        INTEGER*4 IERR, DELTAS(*), B_C
        REAL PHASES(*)
        INTEGER*2 IMAGE_ECHO
        INTEGER*2 BUFFER(32768), IER
        INTEGER*4 STEP, I, J, K, L, L2, ICHAN1, ICHAN2
```

```
      INTEGER*2 FRAMES_PER_SLICE, DATA_BLOCKS, STOP_BLOCK, FRAME_SIZE
      INTEGER*2 NECHOES, FOFF
      INTEGER*2 START_DELTA, STOP_DELTA, NEX_IN_RAW
      REAL ANGLE_ROLL
      REAL*8 DELTA_FACTOR
      REAL*8 PHASE_FACTOR

      INTEGER*4 LN, GET_I

      LOGICAL*2 DO_DELTAS, DO_ROLL, DO_SHIFT

      REAL RR, RI, NEW, SNEW, CNEW

      COMPLEX C1(512)
      REAL B1(2,512)
      EQUIVALENCE(C1,B1)

      COMPLEX C2(512)
      REAL B2(2,512)
      EQUIVALENCE(C2,B2)

      INTEGER*2 START_AT, END_AT

      WRITE(6,2020)

      START_AT = 1
      END_AT = FRAME_SIZE
C...
C...
C...  FIRST, THE HEADER IS COPIED FROM THE SOURCE RAW FILE TO THE
C...  DESTINATION RAW FILE.

      CALL MOVE_HEADER(ICHAN1,ICHAN2,IER)
      IF (IER .NE. 0) THEN
        PRINT*,'ERROR FROM MOVING HEADER, CODE : ',IER
        RETURN
      END IF

C...
C...
C...  FOFF IS AND OFFSET THAT IS USED WHEN DATA ARE READ FROM A
C...  BUFFER. THERE ARE 2 2-BYTE INTEGERS FOR EACH SAMPLE.

      FOFF = FRAME_SIZE * 2

C...
C...
C...  STEP IS THE NUMBER OF 512 BYTE BLOCKS TO READ FOR EACH VIEW.
C...  LN IS USED WITH THE FFT ROUTINE, 2**LN = FRAME_SIZE

      IF (FRAME_SIZE .EQ. 512) THEN
        STEP = 4 * NECHOES * NEX_IN_RAW
        LN = 9
      ELSE IF (FRAME_SIZE .EQ. 256) THEN
        STEP = 2 * NECHOES * NEX_IN_RAW
        LN = 8
      ELSE
        IERR = 4201
        RETURN
```

61

```
          END IF

C...
C...      STOP BLOCK IS THE LAST BLOCK THAT SHOULD BE READ.
C...
          STOP_BLOCK = 28 + STEP * FRAMES_PER_SLICE

          IF (STEP * FRAMES_PER_SLICE .GT. DATA_BLOCKS) THEN
            PRINT*,'ERROR, NOT ENOUGH DATA BLOCKS.'
            IERR = 4200
            RETURN
          END IF

          WRITE(6,2020)

C...
C...      I IS THE BLOCK TO BE READ, IT IS INCREMENTED BY STEP BELOW.
C...      K IS THE VIEW NUMBER CURRENTLY BEING DEALT WITH.
C...
          I = 28
          K = 0
          IERR = 0

C...      THIS LOOP IS DONE ONCE FOR EACH VIEW

          DO WHILE (I .LT. STOP_BLOCK)
            K = K + 1
            WRITE(6,2010)K
2010        FORMAT('+--<GT>',I3,$)
            CALL LIB$RDB (I,STEP,BUFFER,ICHAN1,IERR)
            IF(IERR.NE.0) THEN
              IF ((IERR .NE. -1) .AND. (IERR .NE. 24)) THEN
                PRINT*,' '
                PRINT *, 'ERROR DETECTED WHEN READING BLOCK NUMBER: ',I,' COD
E:',IERR
              ELSE
                PRINT*,' '
                PRINT*,'ERROR, END OF FILE.'
              END IF
              RETURN
            END IF

C...
C...      THIS LOOP IS DONE FOR EACH NEX IN THE RAW DATA. CHOPPER-AP
C...      DATA HAVE ONE NEX IN THE RAW DATA, CHOPPER_NOAP DATA HAVE
C...      TWO OR FOUR, OR MAYBE MORE.
C...
          DO L2 = 0 , NEX_IN_RAW - 1

C...        THE DELTAS ARRAY HAS THE AMOUNT OF SHIFT THAT IS NEEDED.

            IF (DELTAS(K*NEX_IN_RAW-1+L2) .NE. 0) THEN

C...          B_C IS A BUFFER COUNTER THAT IS HOW FAR INTO THE BUFFER
C...          TO BEGIN TO READ FOR THE NEX

              B_C = ((IMAGE_ECHO-1) + L2 * NECHOES) * FOFF
```

```
C...
C...      THE DATA ARE FLIP-FLOPPED, SO THAT SAMPLE 1 GOES TO
C...      POSITION 129, SAMPLE 128 GOES TO POSITION 256, AND
C...      SAMPLE 129 GOES TO POSITION 1. EACH SAMPLE IN THE BUFFER
C...      ARRAY HAS 2 2-BYTES INTEGERS.
C...

          L = FRAME_SIZE/2 + 1
          DO J = 1+B_C , FRAME_SIZE+B_C , 2
             C1(L) = CMPLX(BUFFER(J),BUFFER(J+1))
             L = L + 1
          END DO

          L = 1
          DO J = FRAME_SIZE+1+B_C , FRAME_SIZE*2+B_C , 2
             C1(L) = CMPLX(BUFFER(J),BUFFER(J+1))
             L = L + 1
          END DO

          CALL FFTC(B1(1,1),B1(2,1),LN,2)

          IF (.NOT. DO_DELTAS) THEN
             L = -1
          ELSE
             L = (DELTAS(K*NEX_IN_RAW-1+L2) * DELTA_FACTOR) - 1
          END IF

          J = START_AT - 1

          DO WHILE ((J .LT. END_AT) .AND. (L .LT. 0))
             L = L + 1
             J = J + 1
             C2(FRAME_SIZE+L) = C1(J)
          END DO

          DO WHILE ((J .LT. END_AT) .AND. (L .LT. FRAME_SIZE))
             L = L + 1
             J = J + 1
             C2(L) = C1(J)
          END DO

          DO WHILE (J .LT. END_AT)
             L = L + 1
             J = J + 1
             C2(L-FRAME_SIZE) = C1(J)
          END DO

          IF ((DO_ROLL) .OR. (DO_SHIFT)) THEN

             NEW = 0
             IF (DO_ROLL) THEN
                NEW = NEW + (DELTAS(K*NEX_IN_RAW-1+L2) * DELTA_FACTOR)
     .                      * ANGLE_ROLL
             END IF

             IF (DO_SHIFT) THEN
                NEW = NEW + (PHASES(K*NEX_IN_RAW-1+L2) * PHASE_FACTOR)
```

```
            END IF

            SNEW = SIN(NEW)
            CNEW = COS(NEW)

            DO J = 1 , FRAME_SIZE
              RR = B2(1,J)
              RI = B2(2,J)
              B2(1,J) = RR * CNEW - RI * SNEW
              B2(2,J) = RR * SNEW + RI * CNEW
            END DO

          END IF ! DO_ROLL OR DO_SHIFT

          CALL FFTC(B2(1,1),B2(2,1),LN,-2)

          L = FRAME_SIZE+1+B_C
          DO J = 1 , FRAME_SIZE/2
            BUFFER(L  ) = INT2(AREAL(C2(J)))
            BUFFER(L+1) = INT2(AIMAG(C2(J)))
            L = L + 2
          END DO

          L = 1+B_C
          DO J = FRAME_SIZE/2+1 , FRAME_SIZE
            BUFFER(L  ) = INT2(AREAL(C2(J)))
            BUFFER(L+1) = INT2(AIMAG(C2(J)))
            L = L + 2
          END DO
        END IF
      END DO ! FOR EACH NEX IN RAW
      CALL LIBSWRB (I,STEP,BUFFER,ICHAN2,IERR)
      IF (IERR .NE. 0) THEN
        PRINT*,' '
        PRINT *, 'ERROR DETECTED WHEN WRITTING BLOCK NUMBER: ',I,' CODE
:',IERR
        RETURN
      END IF
      I = I + STEP
      END DO

2020  FORMAT(' ')
      WRITE(6,2020)

      RETURN
      END
```

****
***************** END OF DO_POLO.F77 SOURCE FILE *******************
****

That concludes the four major components of the MARCO and POLO pro-
grams. Below are other source file used in one, some, or all of the
above source file.

****
*************** BEGINNING OF PARCE.F77 SOURCE CODE **************
****

```
      FUNCTION PARSE_I(HOW,NUMS)
      INTEGER*4 PARSE_I,HOW,NUMS(*)
      CHARACTER*80 STR
      LOGICAL*1 NEGATIVE, DONE

      READ(5,1000)STR
1000  FORMAT(A80)

      K = 0
      I = 1
      DONE = .FALSE.
      DO WHILE ((.NOT. DONE) .AND. (K .LT. HOW))
        J = 0

        DO WHILE ((I .LT. 80) .AND. (STR(I:I) .NE. '-') .AND.
     &                              (STR(I:I) .NE. '+') .AND.
     &           ((STR(I:I) .LT. '0') .OR. (STR(I:I) .GT. '9')))
          I = I + 1
        END DO

        IF (I .LT. 80) THEN
          IF (STR(I:I) .EQ. '-') THEN
            NEGATIVE = .TRUE.
            I = I + 1
          ELSE IF (STR(I:I) .EQ. '+') THEN
            NEGATIVE = .FALSE.
            I = I + 1
          ELSE
            NEGATIVE = .FALSE.
          END IF

          IF ((STR(I:I) .GE. '0') .AND. (STR(I:I) .LE. '9')) THEN
            K = K + 1
            DO WHILE ((I .LT. 80) .AND. (STR(I:I) .GE. '0') .AND.
     &               (STR(I:I) .LE. '9'))
              J = J * 10 + (ICHAR(STR(I:I)) - 48)
              I = I + 1
            END DO
          ELSE
            DONE = .TRUE.
          END IF

          IF (NEGATIVE) THEN
            NUMS(K) = J * (-1)
          ELSE
            NUMS(K) = J
          END IF
        ELSE
          DONE = .TRUE.
        END IF
      END DO
      PARSE_I = K
      RETURN
      END
```

65

```fortran
        FUNCTION PARSE_F(HOW,NUMS)
        INTEGER*4 PARSE_F, HOW, K, J
        REAL*8 NUMS(*)
        REAL*8 J2
        CHARACTER*80 STR
        LOGICAL*1 NEGATIVE, DONE

        READ(5,1000)STR
1000    FORMAT(A80)

        K = 0
        I = 1
        DONE = .FALSE.
        DO WHILE ((.NOT. DONE) .AND. (K .LT. HOW))
          J = 0.0

          DO WHILE ((I .LT. 80) .AND. (STR(I:I) .NE. '-') .AND.
     &            (STR(I:I) .NE. '+') .AND. (STR(I:I) .NE. '.') .AND.
     &            ((STR(I:I) .LT. '0') .OR. (STR(I:I) .GT. '9')))
            I = I + 1
          END DO

          IF (I .LT. 80) THEN
            IF (STR(I:I) .EQ. '-') THEN
              NEGATIVE = .TRUE.
              I = I + 1
            ELSE IF (STR(I:I) .EQ. '+') THEN
              NEGATIVE = .FALSE.
              I = I + 1
            ELSE
              NEGATIVE = .FALSE.
            END IF

          IF (((STR(I:I) .GE. '0') .AND. (STR(I:I) .LE. '9'))
     &        .OR. (STR(I:I) .EQ. '.')) THEN
            K = K + 1
            DO WHILE ((I .LT. 80) .AND. (STR(I:I) .GE. '0') .AND.
     &              (STR(I:I) .LE. '9'))
              J = J * 10 + (ICHAR(STR(I:I)) - 48)
              I = I + 1
            END DO
            IF (STR(I:I) .EQ. '.') THEN
              L = I
              I = I + 1
              DO WHILE ((I .LT. 80) .AND. (STR(I:I) .GE. '0') .AND.
     &                (STR(I:I) .LE. '9'))
                J = J * 10 + (ICHAR(STR(I:I)) - 48)
                I = I + 1
              END DO
              M = 2
              J2 = J

              DO WHILE (M .LE. I-L)
                J2 = J2 / 10
                M = M + 1
              END DO
            END IF
```

```
          ELSE
             DONE = .TRUE.
          END IF

          IF (NEGATIVE) THEN
             NUMS(K) =        J2 * (-1)
          ELSE
             NUMS(K) =        J2
          END IF
       ELSE
          DONE = .TRUE.
       END IF
    END DO
    PARSE_F = K
    RETURN
    END


    FUNCTION GET_I
    INTEGER*4 GET_I
    CHARACTER*80 STR
    LOGICAL*1 NEGATIVE

    READ(5,1000) STR
1000 FORMAT(A80)
    I = 1
    J = 0

    DO WHILE ((I .LT. 80) .AND. (STR(I:I) .NE. '-') .AND.
   *                            (STR(I:I) .NE. '+') .AND.
   *          ((STR(I:I) .LT. '0') .OR. (STR(I:I) .GT. '9')))
       I = I + 1
    END DO

    IF (I .LT. 80) THEN
       IF (STR(I:I) .EQ. '-') THEN
          NEGATIVE = .TRUE.
          I = I + 1
       ELSE IF (STR(I:I) .EQ. '+') THEN
          NEGATIVE = .FALSE.
          I = I + 1
       ELSE
          NEGATIVE = .FALSE.
       END IF

       DO WHILE ((I .LT. 80) .AND. (STR(I:I) .GE. '0') .AND.
   *                              (STR(I:I) .LE. '9'))
          J = J * 10 + (ICHAR(STR(I:I)) - 48)
          I = I + 1
       END DO
    END IF

    IF (NEGATIVE) THEN
       J = J * (-1)
    END IF

    GET_I = J
    RETURN
    END
```

67

```
********************************************************************
****************** END OF PARCE.F77 SOURCE CODE *******************
********************************************************************

********************************************************************
*************** BEGINNING OF MAKE_NEW_UDF.F77 SOURCE CODE **********
********************************************************************

!
! COPYRIGHT - MAYO FOUNDATION FOR MEDICAL EDUCATION AND RESEARCH - 1988
!
      SUBROUTINE MAKE_NEW_UDF(FN,IER4)

      INCLUDE ':RDISK:UTIL:QSYM.F77.IN'

      CHARACTER*(*) FN
      INTEGER*4 IER4

      INTEGER*2 CPACK(0:ISYS_CLTH)

      INTEGER*4 AC0, AC1, AC2

      INTEGER*4 CTIM4, CACP4
      EQUIVALENCE (CTIM4,CPACK(ISYS_CTIM))
      EQUIVALENCE (CACP4,CPACK(ISYS_CACP))

      AC0 = BYTEADDR(FN)

      IER4 = ISYS(ISYS_DELETE,AC0,AC1,AC2)

      CPACK(ISYS_CFTYP) = ISYS_ORDY*256 + ISYS_FUDF
      CPACK(ISYS_CCPS)  = 0
      CTIM4 = -1
      CACP4 = -1
      CPACK(ISYS_CDEH)  = 0
      CPACK(ISYS_CDEL)  = -1
      CPACK(ISYS_CMIL)  = -1
      CPACK(ISYS_CMRS)  = 0

      AC0 = BYTEADDR(FN)
      AC2 = WORDADDR(CPACK)

      IER4 = ISYS(ISYS_CREATE,AC0,AC1,AC2)

      RETURN
      END

********************************************************************
**************** END OF MAKE_NEW_UDF.F77 SOURCE FILE **************
********************************************************************
```

68

Claims

We claim:

1. An NMR system, the combination comprising:

   means for generating a polarizing magnetic field;

   excitation means for generating an RF excitation
magnetic field which produces transverse magnetization
5 in nuclei subjected to the polarizing magnetic field;

   receiver means for sensing the NMR signal produced
by the transverse magnetization and producing digitized
in-phase (I) and quadrature (Q) samples of the NMR
signal;

10 first gradient means for generating a first
magnetic field gradient to impart a first phase
component into the NMR signal which is indicative of the
location of the transversely magnetized nuclei along a
first coordinate axis;

15 second gradient means for generating a second
magnetic field gradient to impart a second phase
component into the NMR signal which is indicative of the
location of the transversely magnetized nuclei along a
second coordinate axis;

20   pulse control means coupled to the excitation

means, first and second gradient means, and receiver

means, said pulse control means being operable to

conduct a scan in which a series of pulse sequences are

conducted in which the second magnetic field gradient is

25   stepped through a series of discrete values and a

corresponding series of NMR signals are sensed and

digitized to form an NMR data set; and

processor means for storing the NMR data set and

for reconstructing an image array for a display from the

30   stored NMR data set by:

(a)   Fourier transforming the NMR data set along

one of its dimensions to create hybrid-space data arrays

I' and Q';

(b)   producing a correction data array using the

35   data in the hybrid-space data arrays I' and Q';

(c)   applying the data in the correction data array

to the NMR data set to reduce motion effects; and

(d)   Fourier transforming the corrected NMR image

data set to produce the image array.

2.   The NMR system as recited in claim 1 in which

the correction data array is produced by calculating the

magnitude of the transformed sampled NMR signals in the

hybrid-space data arrays I' and Q' to produce a modulus

5   array, and correlating each row of the modulus array to

produce a corresponding shift value for the correction

data array.

3.   The NMR system as recited in claim 2 in which
a rollback correction data array is produced from the
corresponding shift values in the correction data array
and a rollover rate K_R, and the rollback correction data
5 array values are applied to correct the NMR image data
set in step (c).

4.   The NMR system as recited in claim 1 in which
the correction data array is produced by calculating the
phase of the transformed sampled NMR signals in the
hybrid-space data arrays I' and Q' to produce a two-
5 dimensional phase array, and determining the difference
in phase between elements in a reference row of the
phase array and elements in the same column of the phase
array to produce the values for the correction data
array.

5.   The NMR system as recited in claim 2 in which
a second correction data array is produced by
calculating the phase of the transformed sampled NMR
signals in the hybrid-space data arrays I' and Q' to
5 produce a two-dimensional phase array, and determining
the phase difference between elements in a reference row
of the phase array and elements in the same column of
the phase array to produce the values for the second
correction data array.

6.   The NMR system as recited in claim 1 in which
the NMR data set acquired during the scan includes NMR
navigator data which has been subjected to one of said
two magnetic field gradients and NMR image data which

5   has been subjected to both of said two magnetic field
gradients, and steps (a) and (b) are performed with the
NMR navigator data and step (c) is performed on the NMR
image data.

7.   In an NMR system which performs a scan to
acquire an NMR data set from which an image array is
reconstructed, the improvement comprising:
          transforming an NMR data set produced by the NMR

5   system to create a hybrid-space data array;
          producing a correction data array using the data in
the hybrid-space data array;
          applying the data in the correction data array to
an NMR data set produced by the NMR system to reduce

10   artifacts caused by motion in the acquired NMR data set;
and
          producing an image array from the corrected NMR
data set.

8.   The improvement as recited in claim 7 in which
the hybrid-space data array is created by:
          performing a Fourier transformation on the NMR data
set along one of its dimensions.

9. The improvement as recited in claim 8 in which the correction data array is produced by calculating the magnitude of each element of the hybrid-space data array to produce a corresponding modulus array, and

5 correlating each row of the modulus array to produce a corresponding shift value for the correction data array.

10. The improvement as recited in claim 9 in which a rollback correction data array is produced from the corresponding shift values in the correction data array and a rollover rate K_R, and the values in both the

5 correction data array and the rollback correction data array are applied to correct said acquired NMR data set used to produce the image array.

11. The improvement as recited in claim 8 in which the correction data array is produced by calculating the phase of each element of the hybrid-space data array to produce the corresponding elements of a two-dimensional

5 phase data array, and determining the difference in phase between elements in a reference row of the phase data array and elements in the same column of the phase data array.

12. The improvement as recited in claim 9 in which
a second correction data array is produced by:

calculating the phase of each element of the
hybrid-space data array to produce the corresponding

5   elements of a two-dimensional phase data array having
rows and columns; and

calculating the elements of the second correction
data array by determining the difference in phase
between elements in a reference row of the phase data

10   array and elements in the same column of the phase data
array;

wherein the data in both the correction data array
and the second correction data array is applied to the
acquired NMR data set to reduce the effects of motion.

13. The improvement as recited in claim 7 in which
the NMR data set used to create the hybrid-space data
array is acquired from a navigator NMR signal which is
produced during each pulse sequence of a scan but which

5   is not subjected to a changing magnetic field gradient
during the scan, and the NMR data set employed to
produce the image array is acquired from an NMR signal
which is produced during each pulse sequence of the scan
and which is subject to a phase encoding magnetic field

10   gradient which changes during the scan.

14. The improvement as recited in claim 13 in which each navigator NMR signal is produced in the same pulse sequence with its corresponding phase encoded NMR signal.

15. An NMR system, the combination comprising:
means for generating a polarizing magnetic field;
excitation means for generating an RF excitation magnetic field which produces transverse magnetization

5 in nuclei subjected to the polarizing magnetic field;
receiver means for sensing the NMR signal produced by the transverse magnetization and producing digitized in-phase (I) and quadrature (Q) samples of the NMR signal;

10 first gradient means for generating a first magnetic field gradient to impart a first phase component into the NMR signal which is indicative of the location of the transversely magnetized nuclei along a first coordinate axis;

15 second gradient means for generating a second magnetic field gradient to impart a second phase component into the NMR signal which is indicative of the location of the transversely magnetized nuclei along a second coordinate axis;

75

20   pulse control means coupled to the excitation

means, first and second gradient means, and receiver

means, said pulse control means being operable to

conduct a scan in which a series of pulse sequences are

conducted in which the second magnetic field gradient is

25   stepped through a series of discrete values and a

corresponding series of NMR signals are sensed and

digitized to form an NMR data set; and

processor means for storing the NMR data set and

for reconstructing an image array for a display from the

30   stored NMR data set by:

(a) producing a phase array from the NMR data set

which indicates the phase of the digitized NMR signals;

(b) producing correction data using the data in

the phase array;

35   (c) applying the correction data to the NMR data

set to reduce motion effects; and

(d) Fourier transforming the corrected NMR image

data set to produce the image array.

16. An NMR system, the combination comprising:

means for generating a polarizing magnetic field;

excitation means for generating an RF excitation
magnetic field which produces transverse magnetization

5 in nuclei subjected to the polarizing magnetic field;

receiver means for sensing the NMR signal produced
by the transverse magnetization and producing digitized
in-phase (I) and quadrature (Q) samples of the NMR
signal;

10 first gradient means for generating a first
magnetic field gradient to impart a first phase
component into the NMR signal which is indicative of the
location of the transversely magnetized nuclei along a
first coordinate axis;

15 second gradient means for generating a second
magnetic field gradient to impart a second phase
component into the NMR signal which is indicative of the
location of the transversely magnetized nuclei along a
second coordinate axis;

20 pulse control means coupled to the excitation
means, first and second gradient means, and receiver
means, said pulse control means being operable to
conduct a scan in which a series of pulse sequences are
conducted in which the second magnetic field gradient is

25 stepped through a series of discrete values and a
corresponding series of NMR signals are sensed and
digitized to form an NMR data set; and

processor means for storing the NMR data set and
for reconstructing an image array for a display from the
30   stored NMR data set by:

(a)   producing a phase array from the NMR data set
which indicates the phase of the digitized NMR signals;

(b)   producing an altered phase array by
subtracting phase components produced by stationary
35   spins from each value in the phase array;

(c)   producing correction data using the data in
the altered phase array;

(d)   applying the correction data to the NMR data
set to reduce motion effects; and

40   (e)   transforming the corrected NMR image data set
to produce the image array.

17.   The NMR system as recited in claim 16 in which
the phase components produced by stationary spin are
calculated by determining the center of a circle which
best fits the values in each column of the phase array.
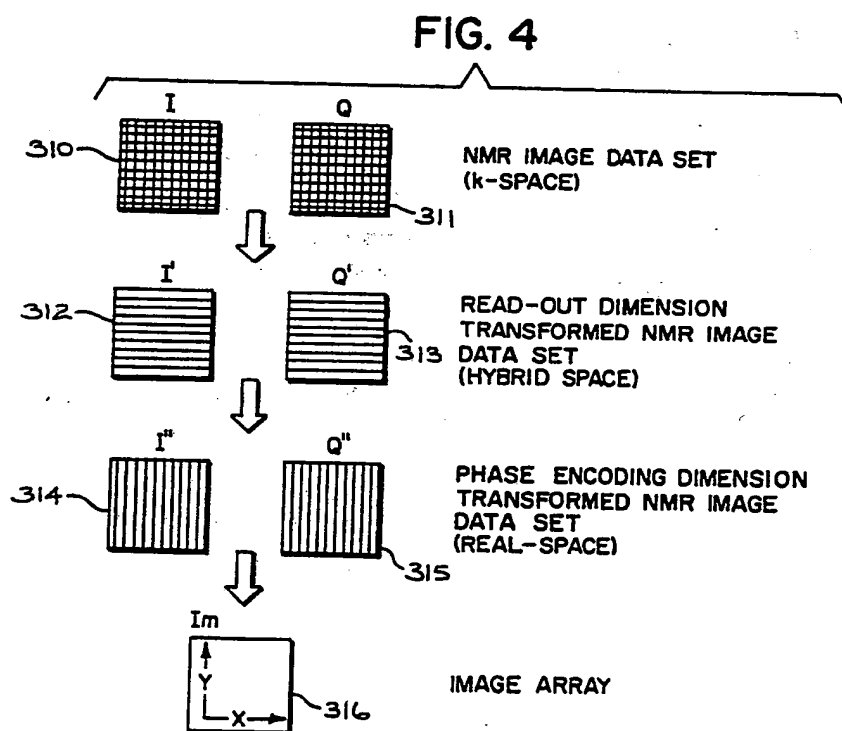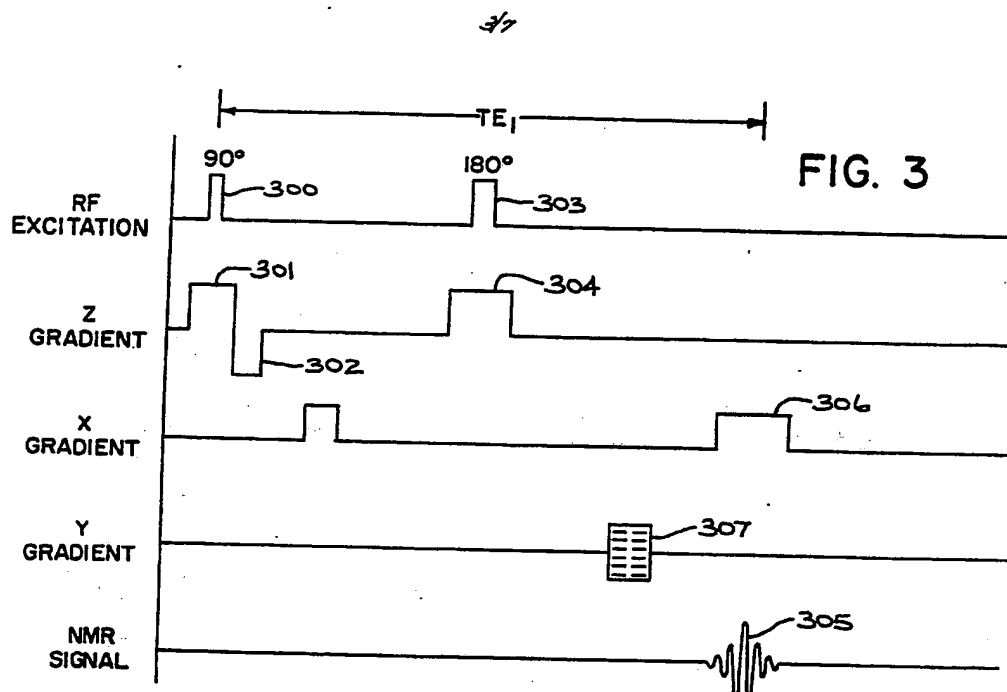
FIG. 1

2/7

FIG. 2

FIG. 3

FIG. 4

4/7

I'   Q'
312   NMR DATA
                                         (HYBRID–SPACE)
                                                    313

φ   M   320

PHASE ARRAY   MODULUS ARRAY
(HYBRID–SPACE)   (HYBRID–SPACE)
                    335

PHASE   SHIFT
CORRECTION   330   CORRECTION
ARRAY (P)   ARRAY (S)
                    337

                                         ROLLBACK
                                 331   CORRECTION
                                         ARRAY (R)

FIG. 5

FIG. 6

5/7

FIG. 7

FIG. 8A

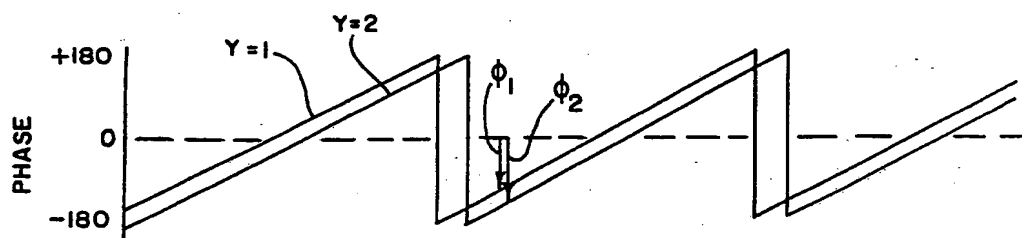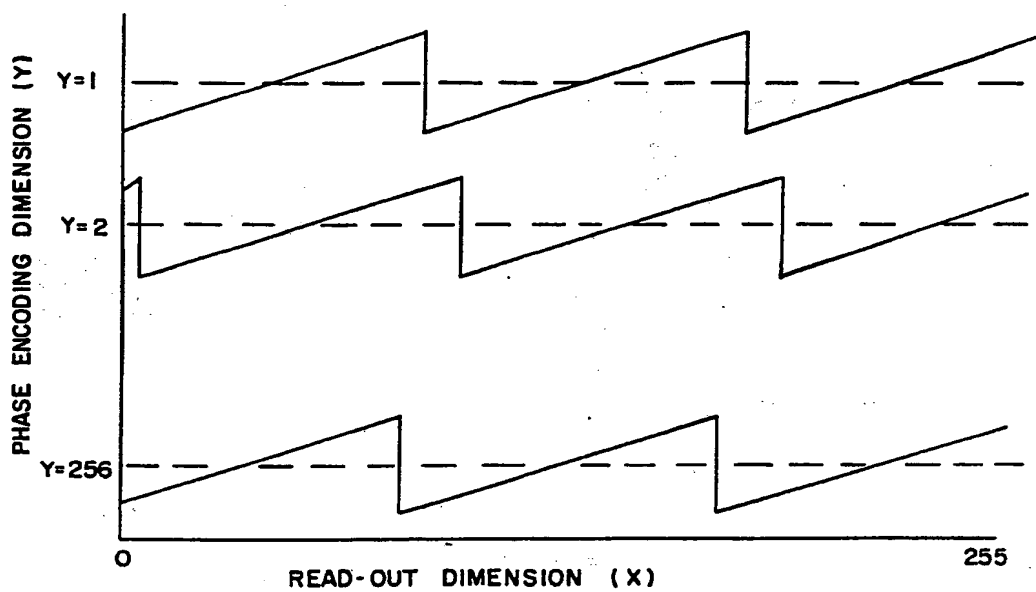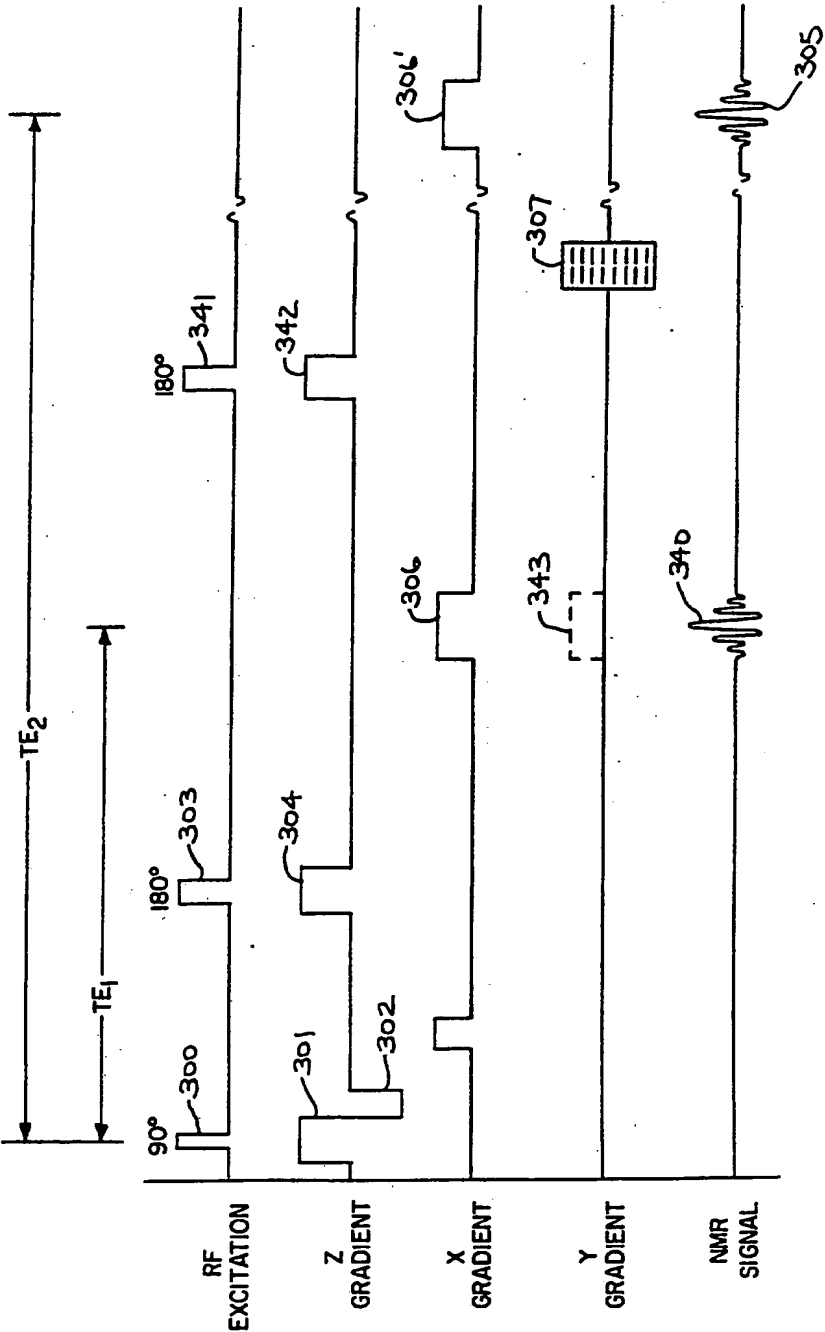PHASE ENCODING DIMENSION (Y)

READ-OUT DIMENSION (X)
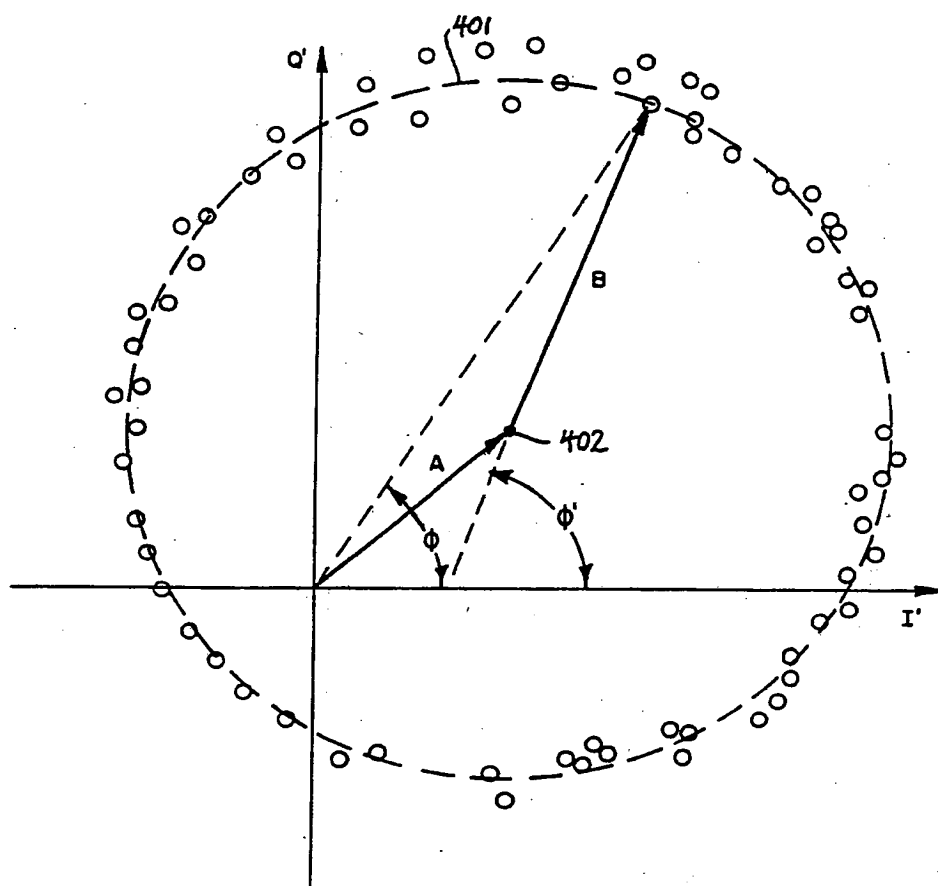

FIG. 8B

SUBSTITUTE SHEET
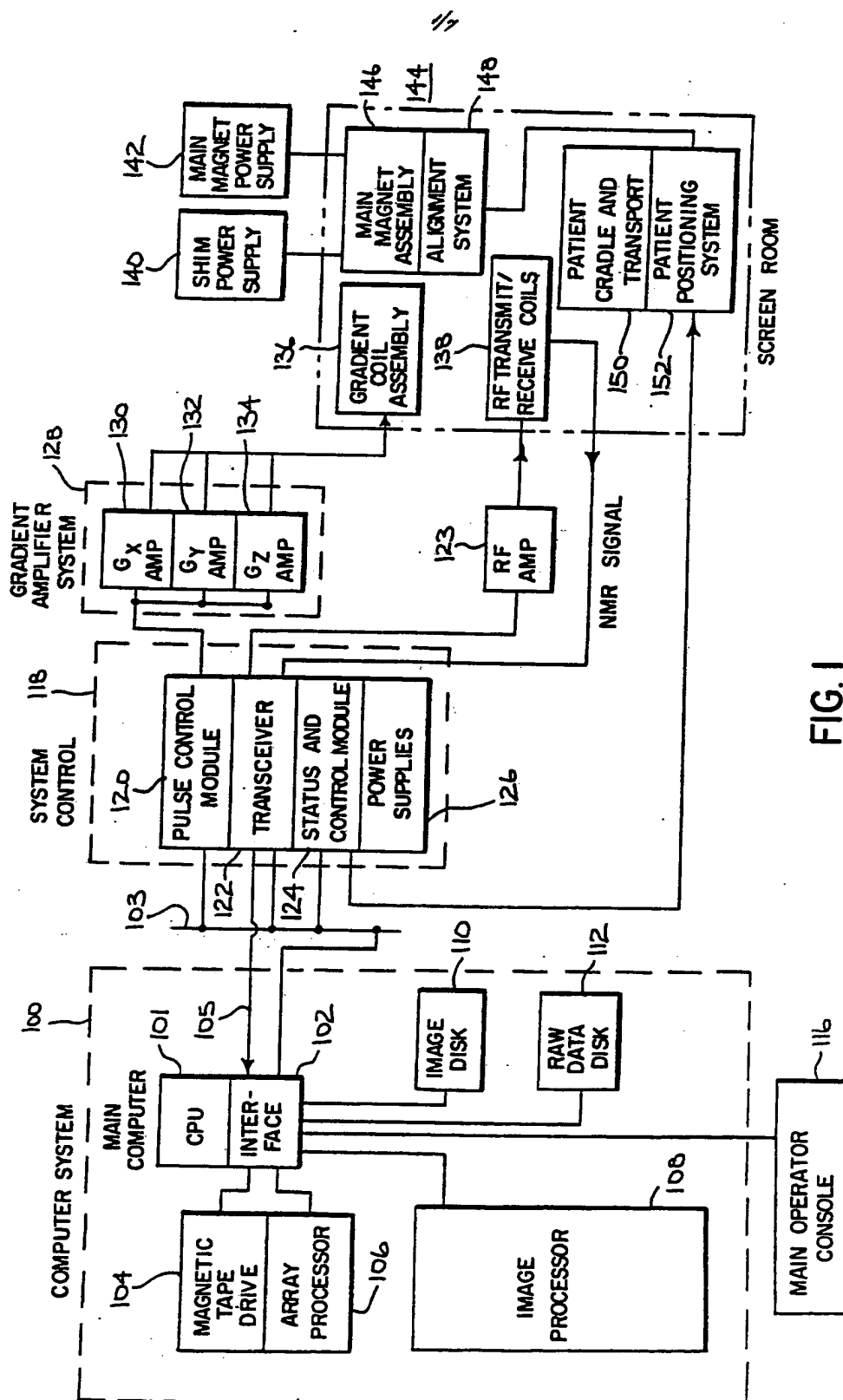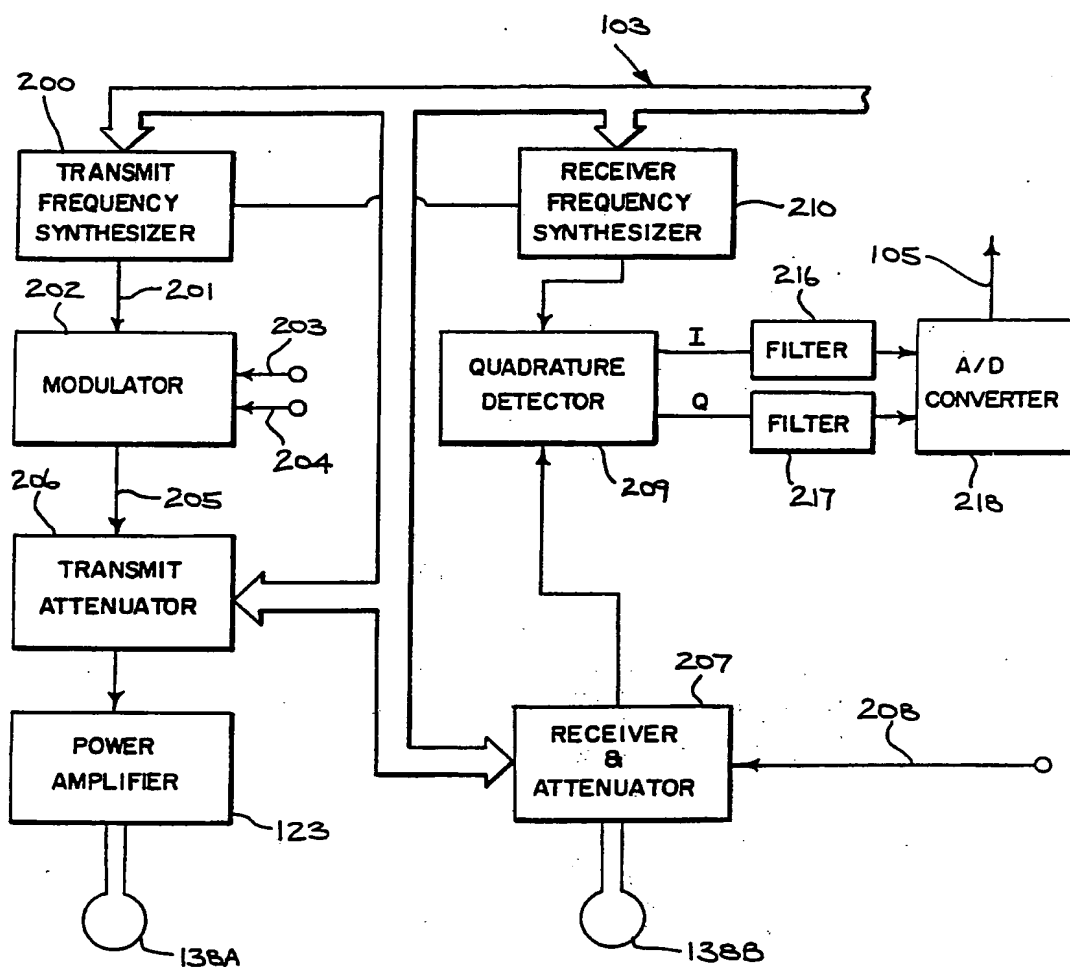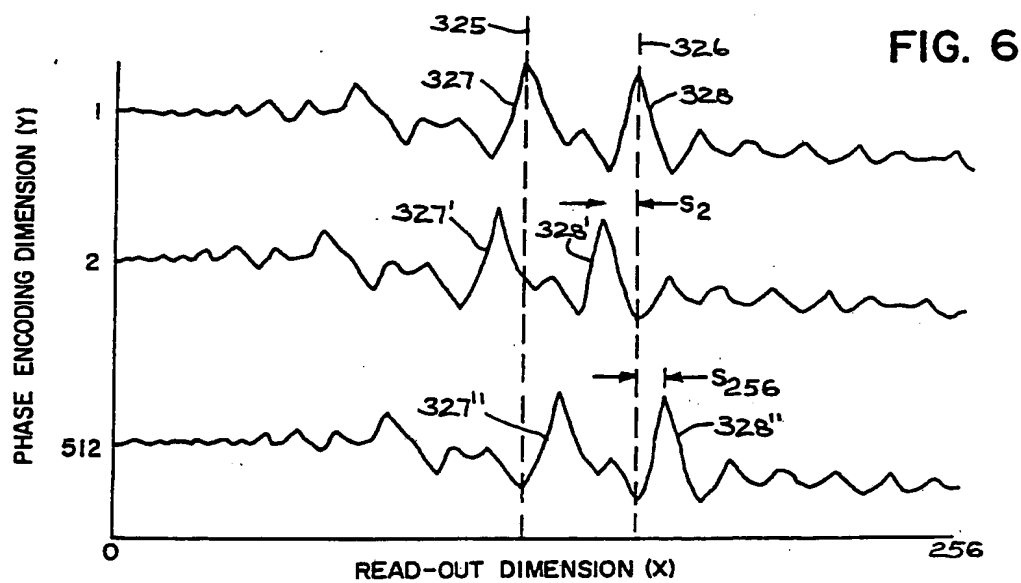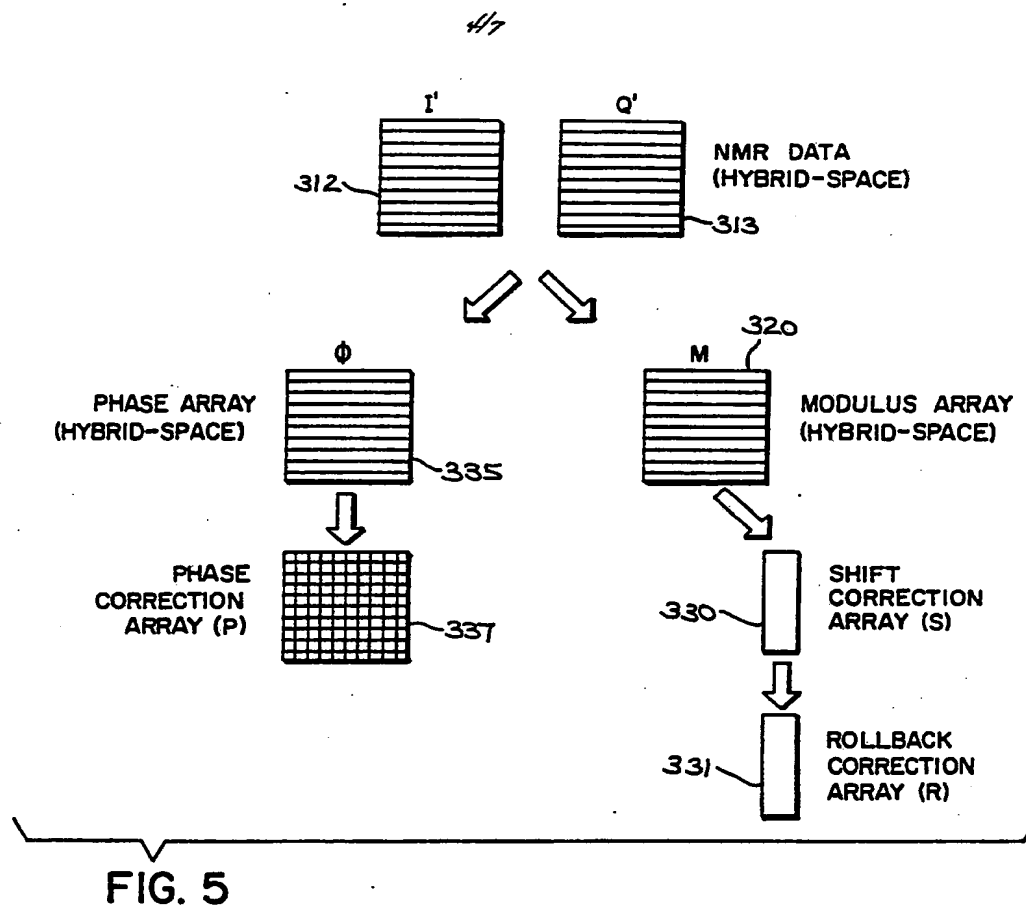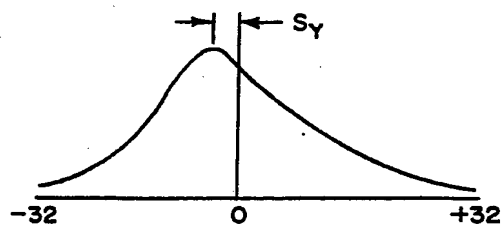
FIG. 9

FIG. 10

INTERNATIONAL SEARCH REPORT

International Application No. PCT/US89/04945

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) 6

According to International Patent Classification (IPC) or to both National Classification and IPC

IPC (5) G01R 33/20
U S CL 324/300, 307, 309, 318, 322

II. FIELDS SEARCHED

Minimum Documentation Searched 7

Classification System	Classification Symbols
U S	324/ 300, 307, 309, 318, 322 ·

Documentation Searched other than Minimum Documentation
to the Extent that such Documents are Included in the Fields Searched 8

III. DOCUMENTS CONSIDERED TO BE RELEVANT 9

Category *	Citation of Document, 11 with indication, where appropriate, of the relevant passages 12	Relevant to Claim No. 13
A	US, A, 4,706, 026, (Pelc), 10 November, 1987	1-17
A	US, A, 4,715,383, (Ehman), 29 December, 1987	1-17
A	US, A, 4,731,583, (Glover), 15 March, 1988	1-17

* Special categories of cited documents: 10

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search	Date of Mailing of this International Search Report
29 January 1990	1 3FEB 1990

International Searching Authority	Signature of Authorized Officer
ISA/US	Michael J. Tokar

Form PCT/ISA/210 (second sheet) (Rev.11-87)

THIS PAGE BLANK (USPTO)